Manual do "SMART PIC II"

Manual de Utilização do SMART PIC

Capítulos:

1. Instalando o MPLAB – Software de desenvolvimento de projetos 3
2. Instalando o Programador SmartPic Software de gravação do PIC 16
3. Abrindo um novo projeto 17
   - 3.1 - Abrindo um novo projeto para linguagem assembler 17
   - 3.2 - Criando um Novo Programa 24
   - 3.3 - Compilando o programa 29
4. Pinagem no PIC 16F873/A e 16F84/A 30
5. Conhecendo os mneumônicos do PIC – Apenas 35 instruções para aprender 31
6. Gravando o programa no microcontrolador PIC 54
7. Conhecendo a placa 57
   - 7.1 - Configuração das PORTAS de entrada e saída para o PIC 16F84/A. 58
   - 7.2 - Configuração das PORTAS de entrada e saída para o PIC 16F873/A. 58
   - 7.3 - Conector CN2 (conector de entradas remotas) 59
   - 7.4 - Conector CN3 (conector de saídas remotas e comunicação serial) 59
   - 7.5 - Configuração dos Jumpers 60
   - 7.6 – Conexão do KIT com o computador 61

Manual de Utilização do SMART PIC

Capítulo 1

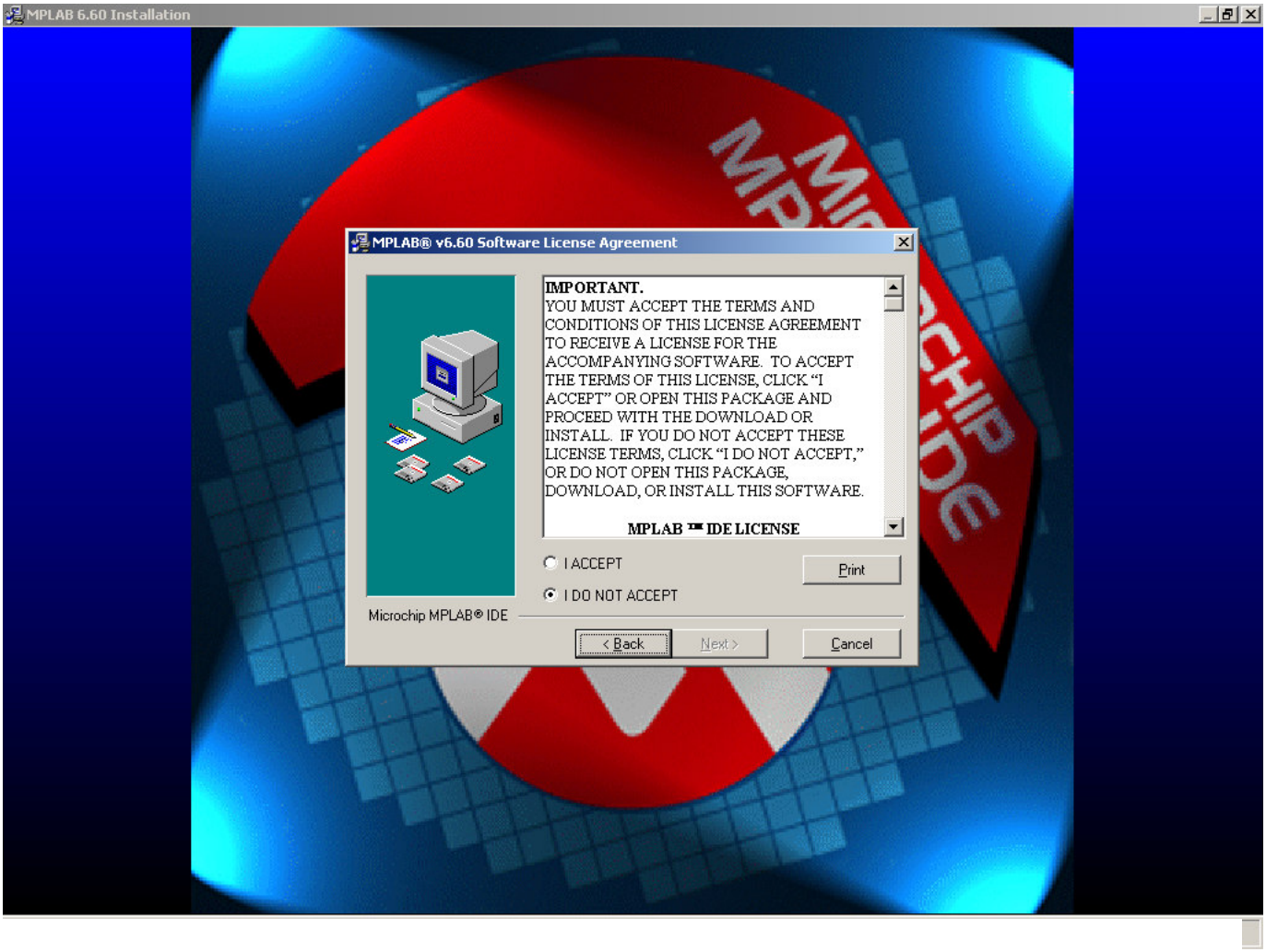
1-Instalando o MPLAB

1.1 – Abra a CD no diretório MPLAB, e clique no arquivo MPLAB66000.EXE.

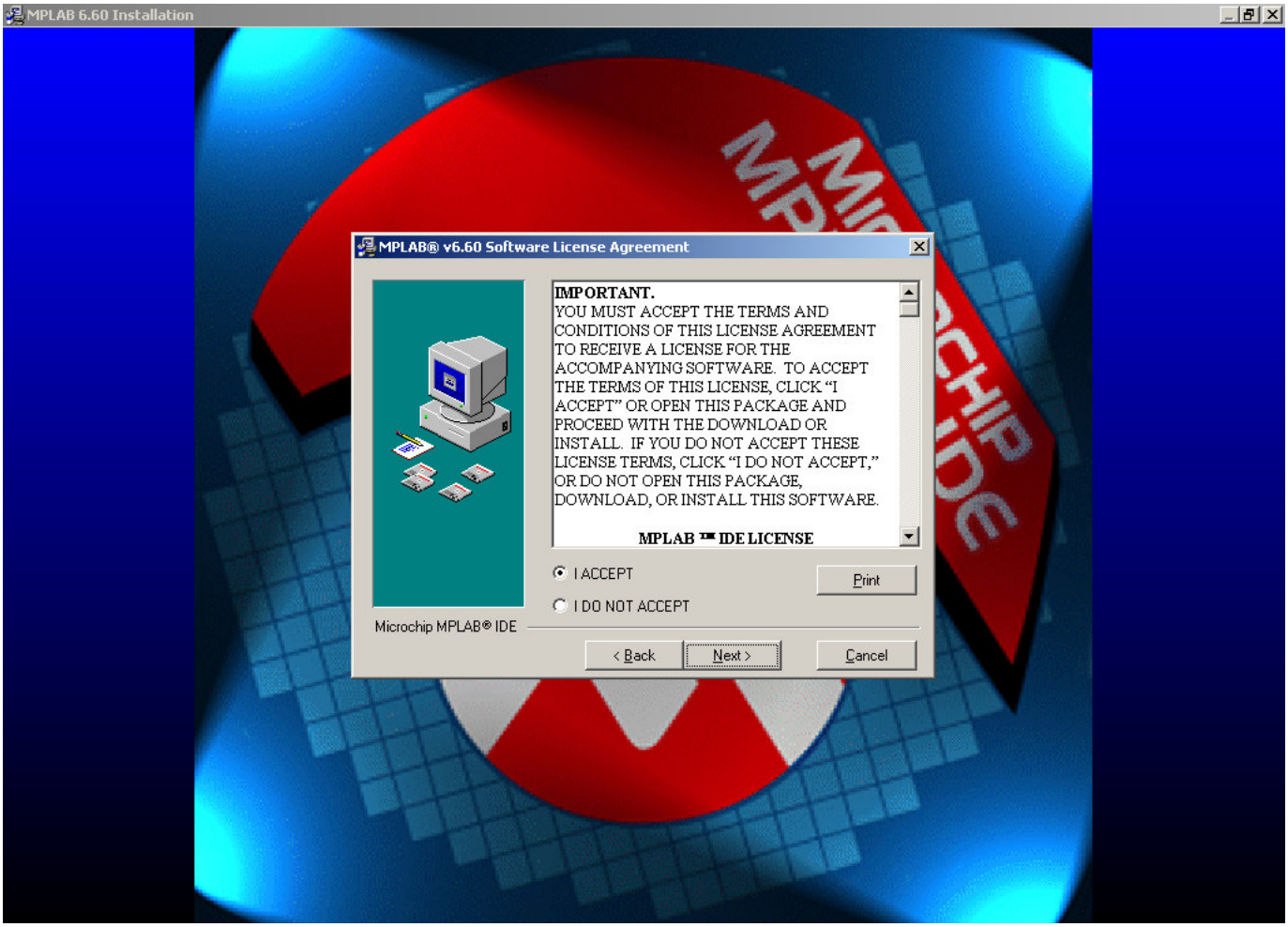
MPLAB 6.60 Installation
MPLAB® v6.60 Installation on Windows 2000
Welcome!
This installation program will install the MPLAB® v6.60.
It is preferable to run this installation program from a re-boot prior to running any other applications. This will ensure the minimum amount of system dlls that may be loaded into memory.
You may try to close all other applications before continuing but it is not guaranteed that some dlls may not already be loaded into memory.
There are system dlls that may require updating, and can not be replaced if in use.
Press the Next button to start the installation. You can press the Cancel button now if you do not want to install the MPLAB® v6.60 at this time.
Microchip MPLAB® IDE
Next >
Cancel

Manual de Utilização do SMART PIC

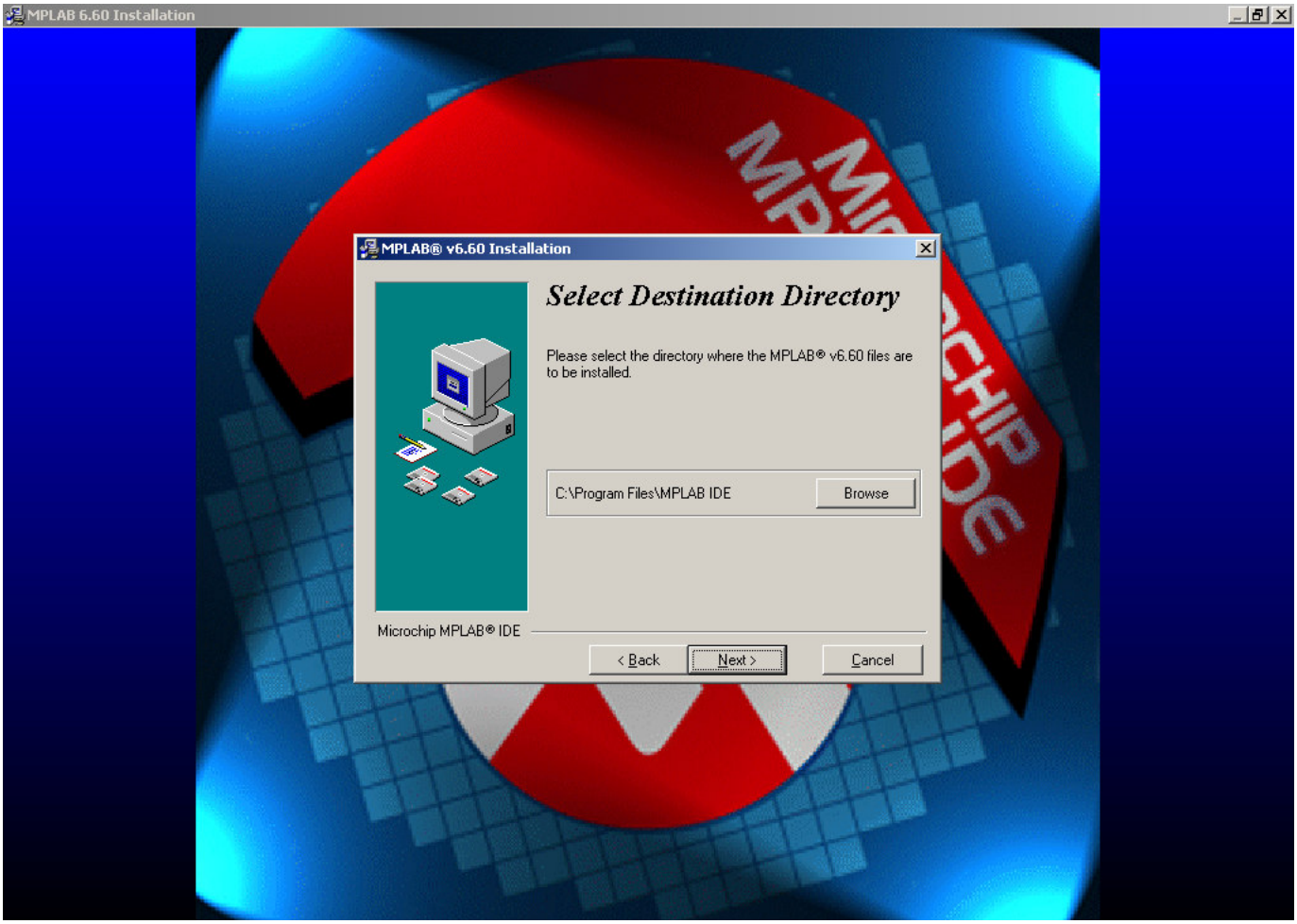
1.2 clique em NEXT


MPLAB 6.60 Installation
MPLAB® v6.60 Software License Agreement
IMPORTANT.
YOU MUST ACCEPT THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT TO RECEIVE A LICENSE FOR THE ACCOMPANYING SOFTWARE. TO ACCEPT THE TERMS OF THIS LICENSE, CLICK "I ACCEPT" OR OPEN THIS PACKAGE AND PROCEED WITH THE DOWNLOAD OR INSTALL. IF YOU DO NOT ACCEPT THESE LICENSE TERMS, CLICK "I DO NOT ACCEPT," OR DO NOT OPEN THIS PACKAGE, DOWNLOAD, OR INSTALL THIS SOFTWARE.
MPLAB IDE LICENSE
I ACCEPT
I DO NOT ACCEPT
Print
Microchip MPLAB® IDE
< Back
Next >
Cancel

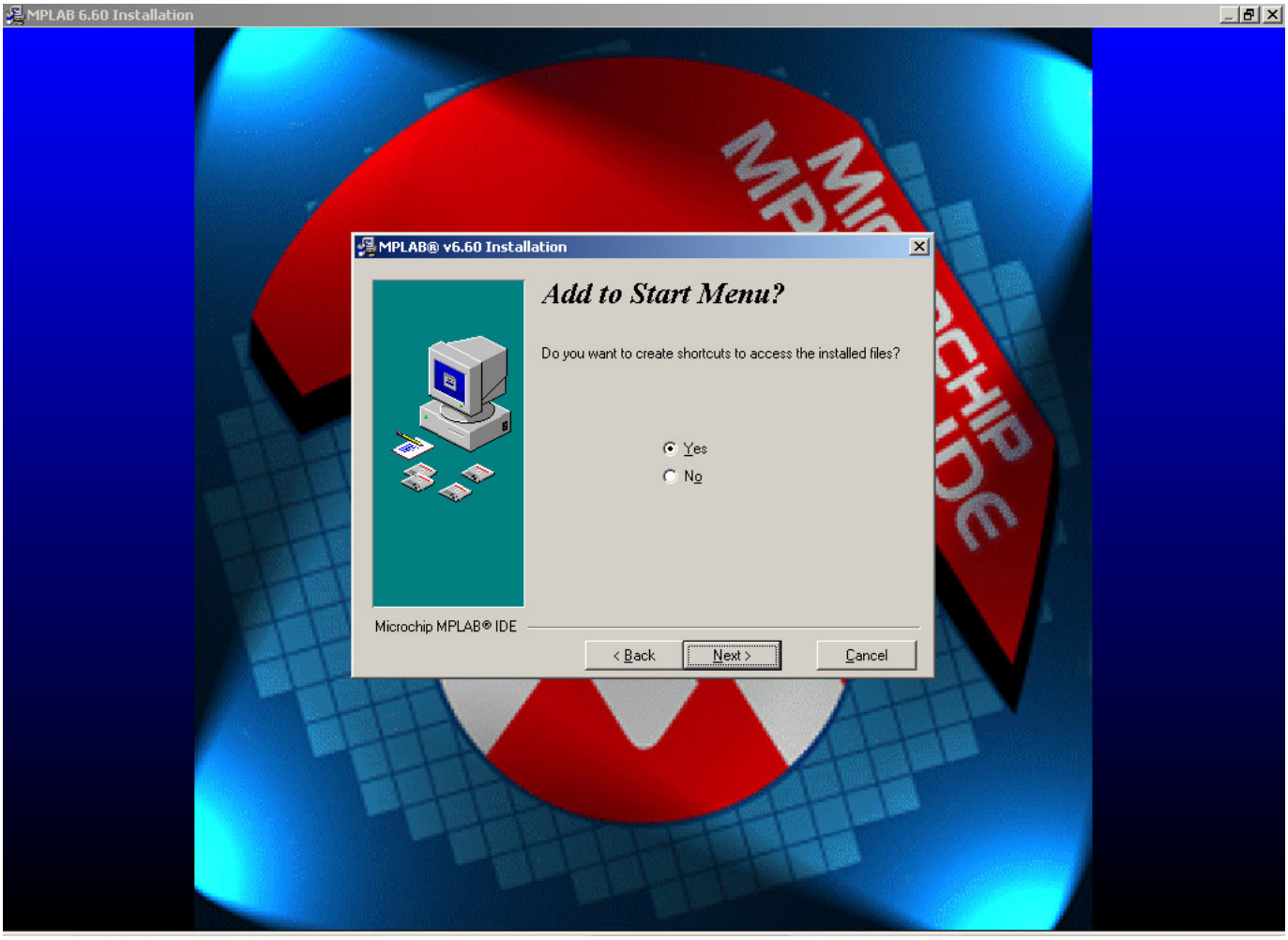
1.3 - Selecione I AGREE, e clique em NEXT


MPLAB 6.60 Installation
MPLAB® v6.60 Software License Agreement
IMPORTANT.
YOU MUST ACCEPT THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT TO RECEIVE A LICENSE FOR THE ACCOMPANYING SOFTWARE. TO ACCEPT THE TERMS OF THIS LICENSE, CLICK "I ACCEPT" OR OPEN THIS PACKAGE AND PROCEED WITH THE DOWNLOAD OR INSTALL. IF YOU DO NOT ACCEPT THESE LICENSE TERMS, CLICK "I DO NOT ACCEPT," OR DO NOT OPEN THIS PACKAGE, DOWNLOAD, OR INSTALL THIS SOFTWARE.
MPLAB IDE LICENSE
I ACCEPT
I DO NOT ACCEPT
Print
Microchip MPLAB® IDE
< Back
Next >
Cancel

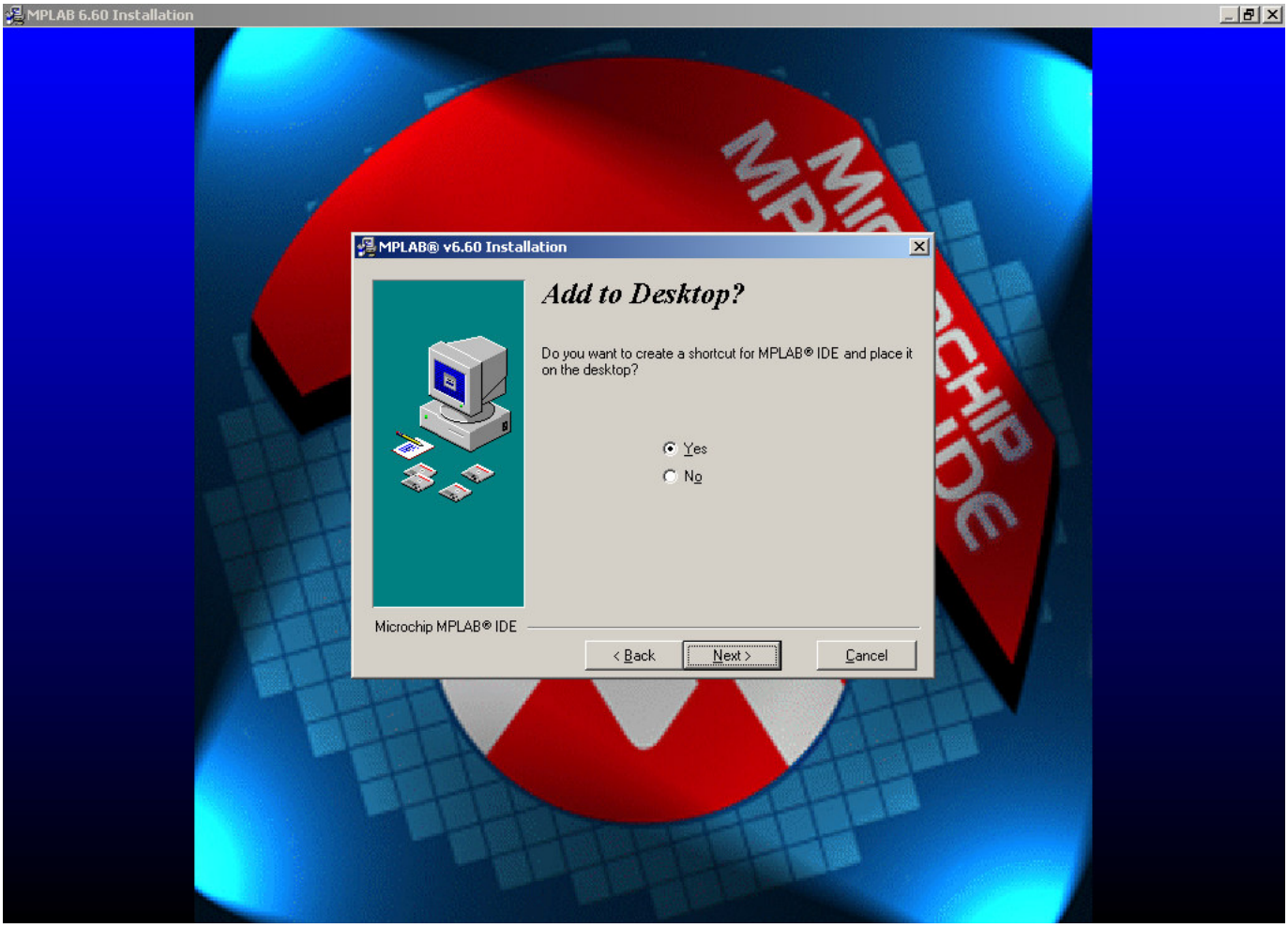
1.4 – Selecione o diretório que será instalado o MPLAB



1.5 – Se você desejar criar atalhos para o MPLAB selecione YES.

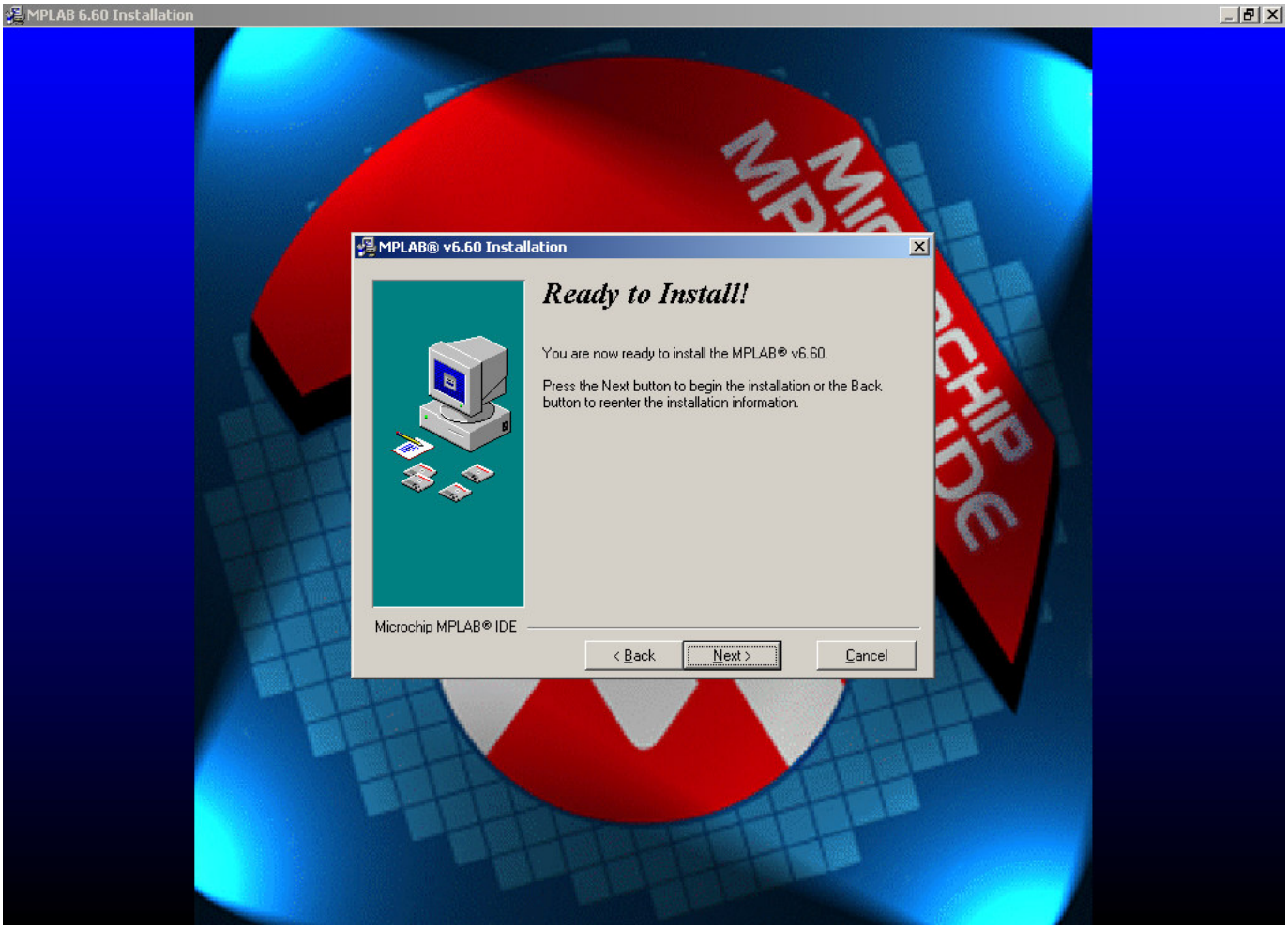


1.6 – Caso você deseje que o ícone do MPLAB apareça no desktop selecione YES.

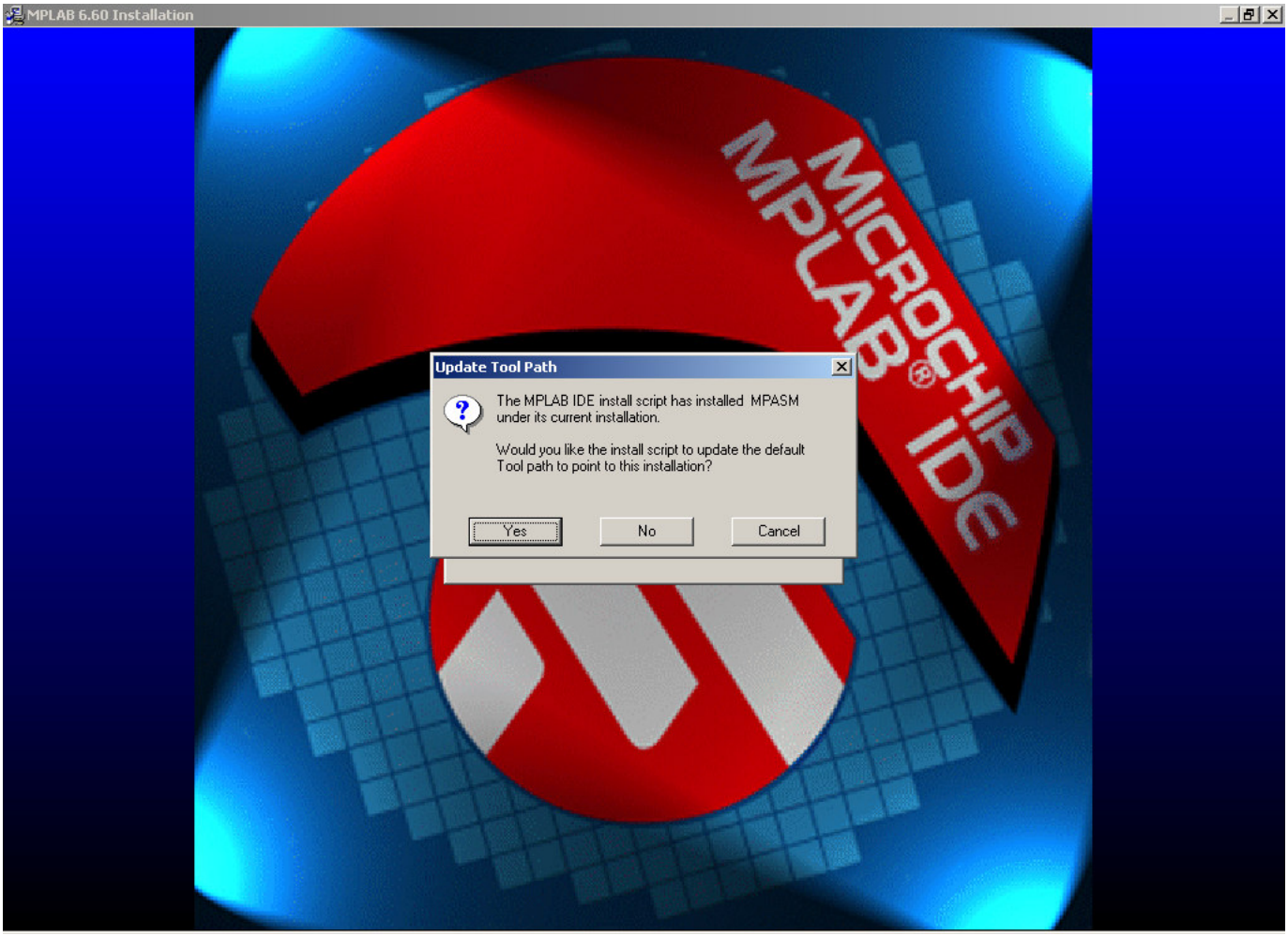


Manual de Utilização do SMART PIC

1.7 – Se você estiver pronto para instalar clique em Next, caso contrário de Back e altere as opções pré selecionadas.


MPLAB 6.60 Installation
MPLAB® v6.60 Installation
Ready to Install!
You are now ready to install the MPLAB® v6.60.
Press the Next button to begin the installation or the Back button to reenter the installation information.
Microchip MPLAB® IDE
< Back
Next >
Cancel

1.8 – Se você tiver outras instalações do MPLAB,e quiser que esta instalação seja a padrão clique em YES até o update de register.


MPLAB 6.60 Installation
MICROCHIP
MPLAB® IDE
Update Tool Path
The MPLAB IDE install script has installed MPASM under its current installation.
Would you like the install script to update the default Tool path to point to this installation?
Yes
No
Cancel

Manual de Utilização do SMART PIC

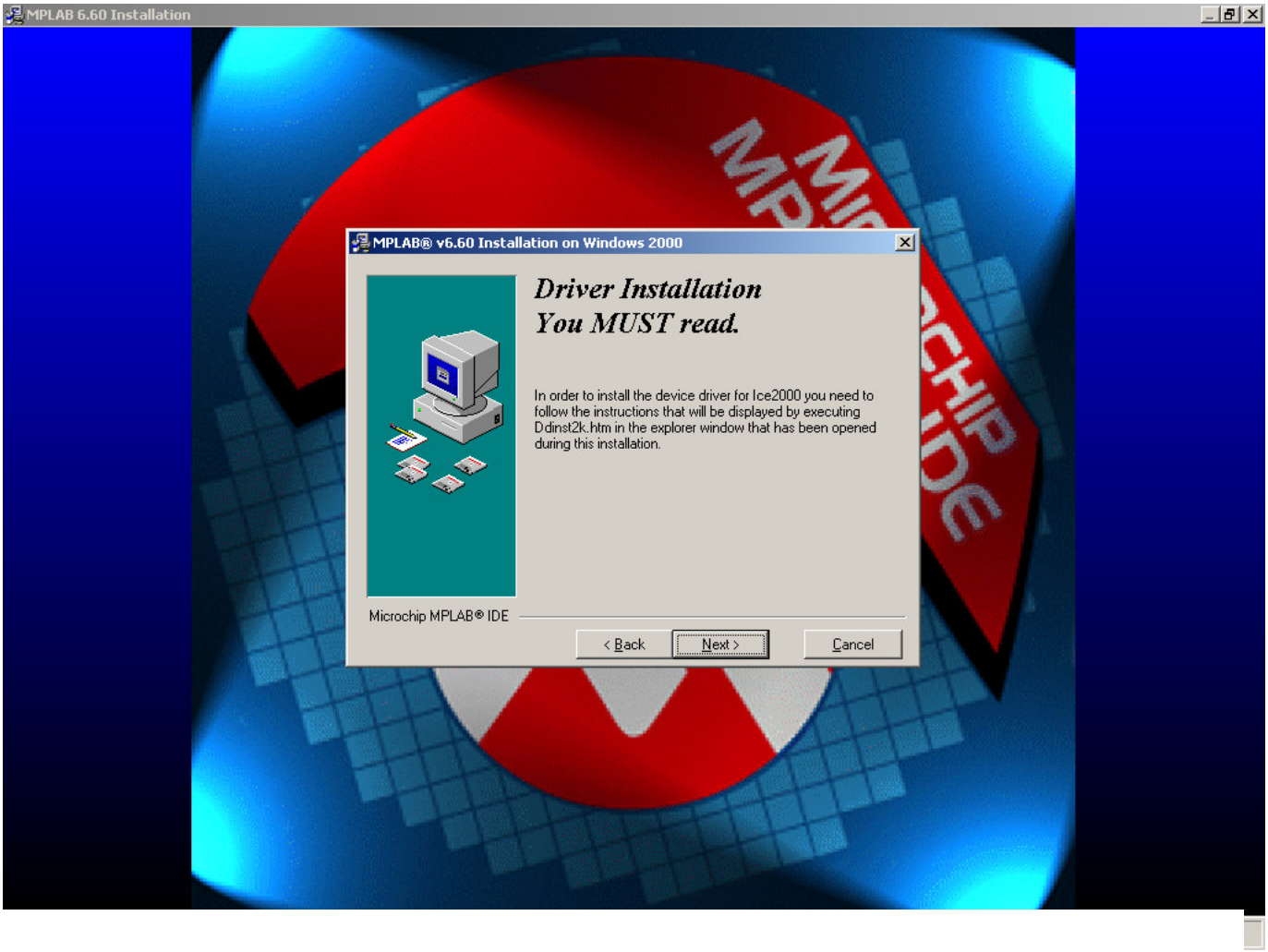
1.9 – Se você quiser ler os readme files clique em YES, caso contrário selecione NO.

MPLAB 6.60 Installation
MPLAB® v6.60 Installation
View README Files?
MPLAB IDE has associated README files that contains important information, such as device support and known issues.
Would you like to view these files now?
Yes
No
Please review these files.
Microchip MPLAB® IDE
Next >
Cancel

1.10. – Clique em YES.

MPLAB 6.60 Installation
MPLAB® v6.60 Installation on Windows 2000
USB Driver Installation
Please read.
In order to install the USB device driver for MPLAB ICD2 you need to follow the instructions that will be displayed by executing ezicd2.htm in the explorer window that has been opened during this installation.
The installation is automatic when the MPLAB ICD2 is connected to the USB port. The instructions are for reference.
Microchip MPLAB® IDE
Next >
Cancel

1.11 Clique em NEXT.

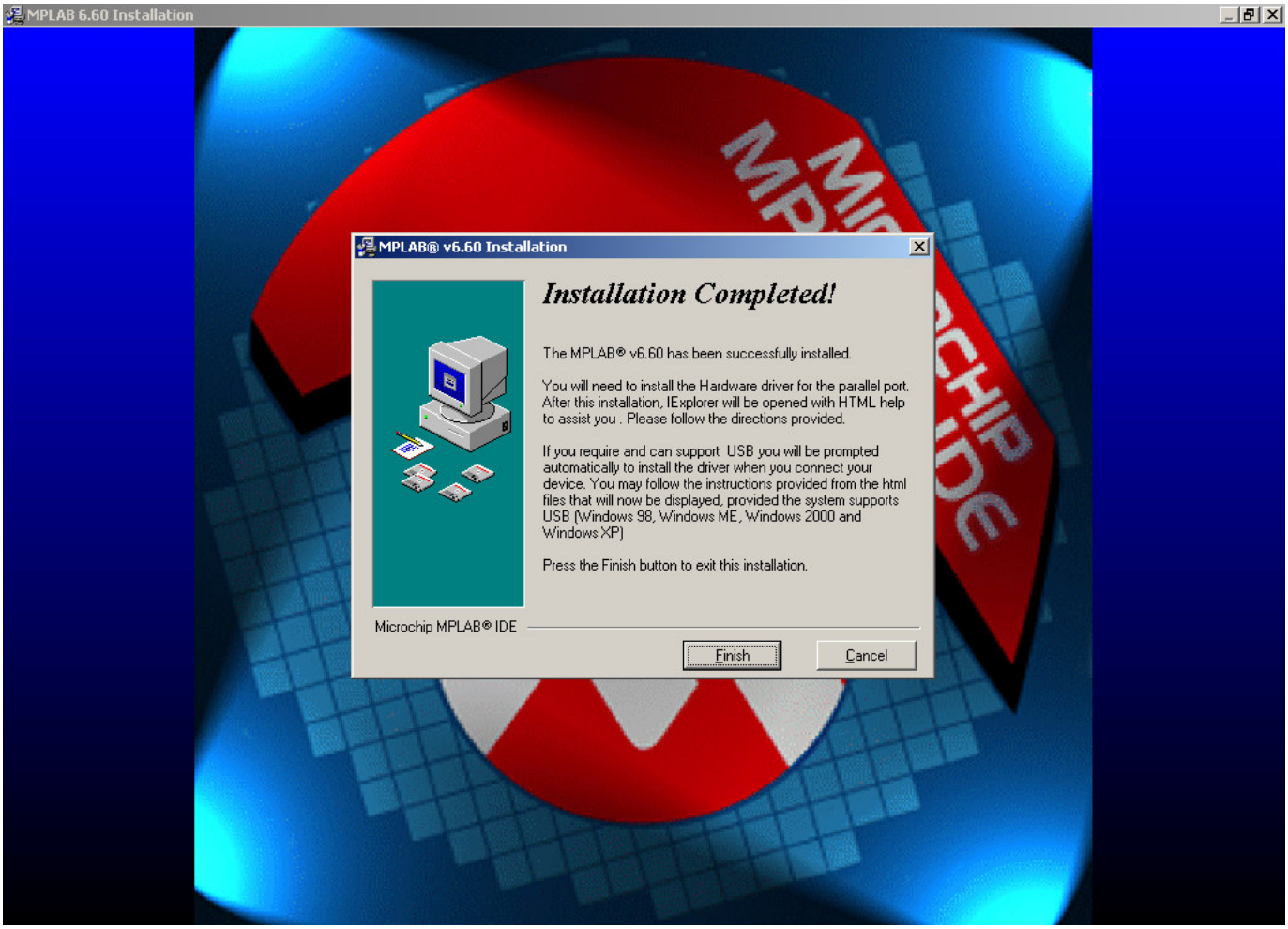

MPLAB 6.60 Installation
MPLAB® v6.60 Installation on Windows 2000
Driver Installation
You MUST read.
In order to install the device driver for Ice2000 you need to follow the instructions that will be displayed by executing Ddinst2k.htm in the explorer window that has been opened during this installation.
Microchip MPLAB® IDE
< Back
Next >
Cancel

Manual de Utilização do SMART PIC

1.12 – Clique em NEXT, clique em NEXT.

MPLAB 6.60 Installation
MPLAB® v6.60 Installation on Windows 2000
USB Driver Installation
Please read.
In order to install the USB device driver for MPLAB ICE 4000 you need to follow the instructions that will be displayed by executing ezice4k.htm in the explorer window that has been opened during this installation.
The installation is automatic when the MPLAB ICE 4000 is connected to the USB port. The instructions are for reference.
Microchip MPLAB® IDE
< Back
Next
Cancel

1.13 – Instalação OK, clique em FINISH.

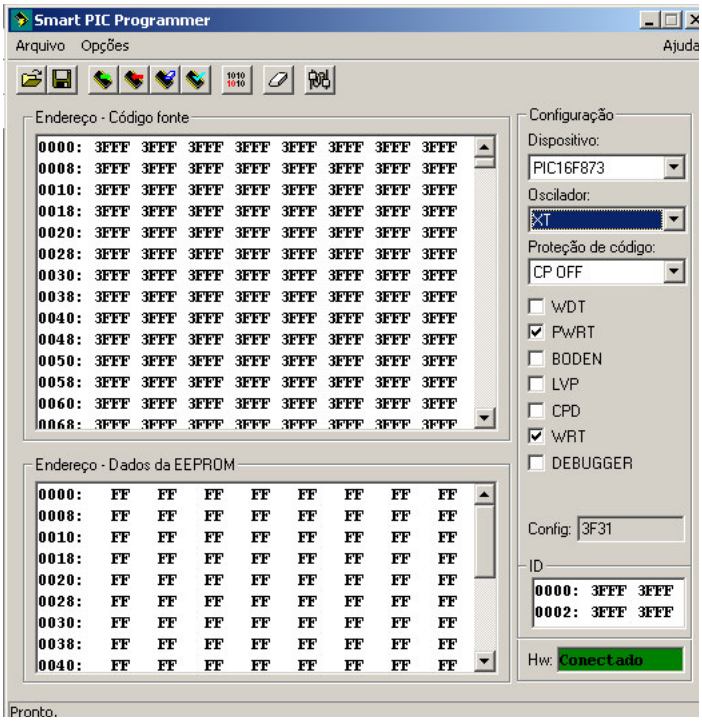

MPLAB 6.60 Installation
MPLAB® v6.60 Installation
Installation Completed!
The MPLAB® v6.60 has been successfully installed.
You will need to install the Hardware driver for the parallel port. After this installation, IExplorer will be opened with HTML help to assist you . Please follow the directions provided.
If you require and can support USB you will be prompted automatically to install the driver when you connect your device. You may follow the instructions provided from the html files that will now be displayed, provided the system supports USB (Windows 98, Windows ME, Windows 2000 and Windows XP)
Press the Finish button to exit this installation.
Microchip MPLAB® IDE
Finish
Cancel

Manual de Utilização do SMART PIC

2 - Instalando o Programador SmartPic software de gravação do PIC.

Copie todos os arquivos do CD no diretório ProgramadorSmartPic para seu hard disk (deve ter 4 arquivos). Se você não visualizar o arquivos mude as opções de visualização dos arquivos do EXPLORER.
Dê duplo clique no arquivo Programadorsmartpic.exe deve aparecer a imagem abaixo.

Não esqueça de selecionar o dispositivo coerente.



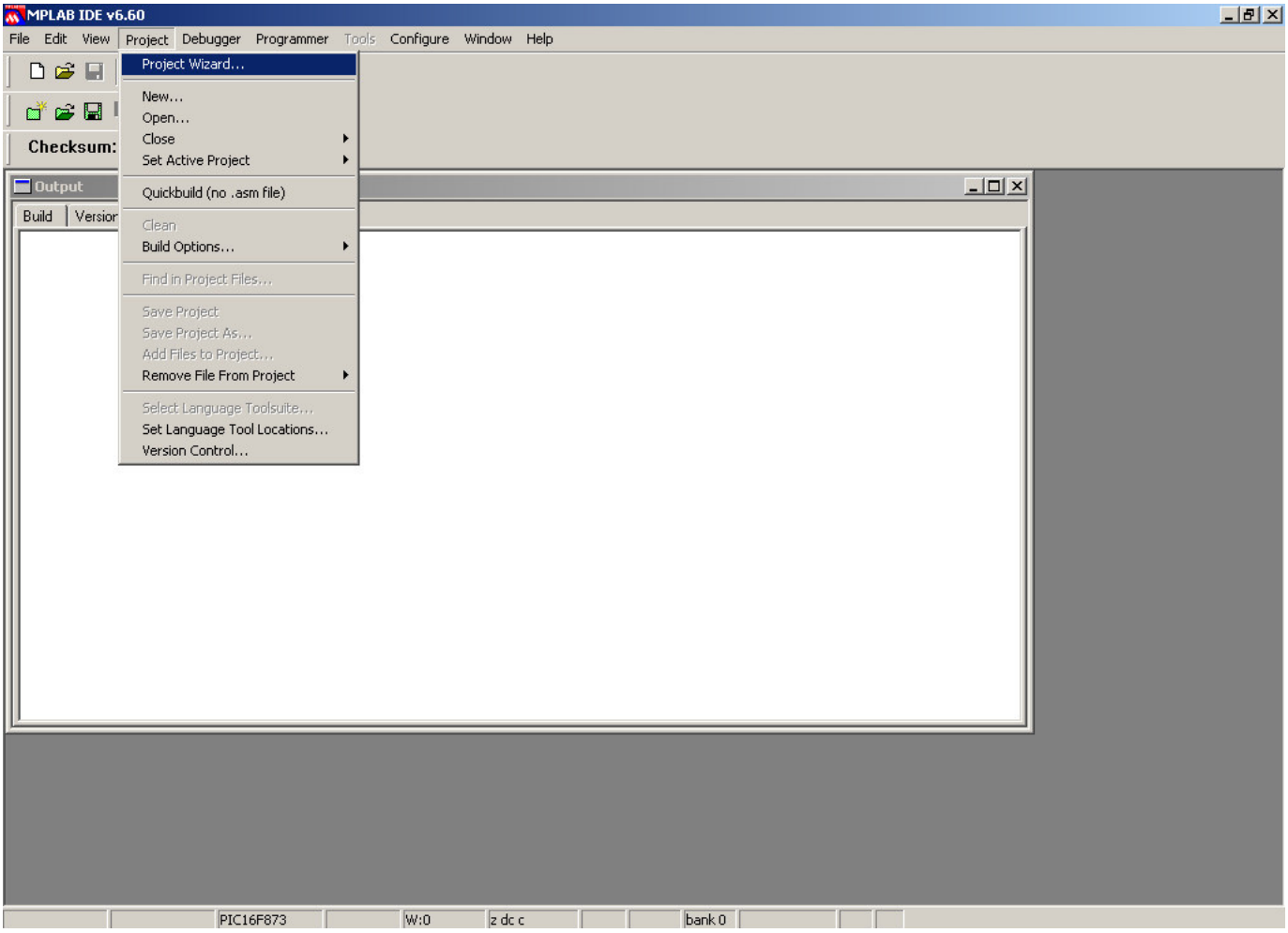
Manual de Utilização do SMART PIC

Capítulo 3

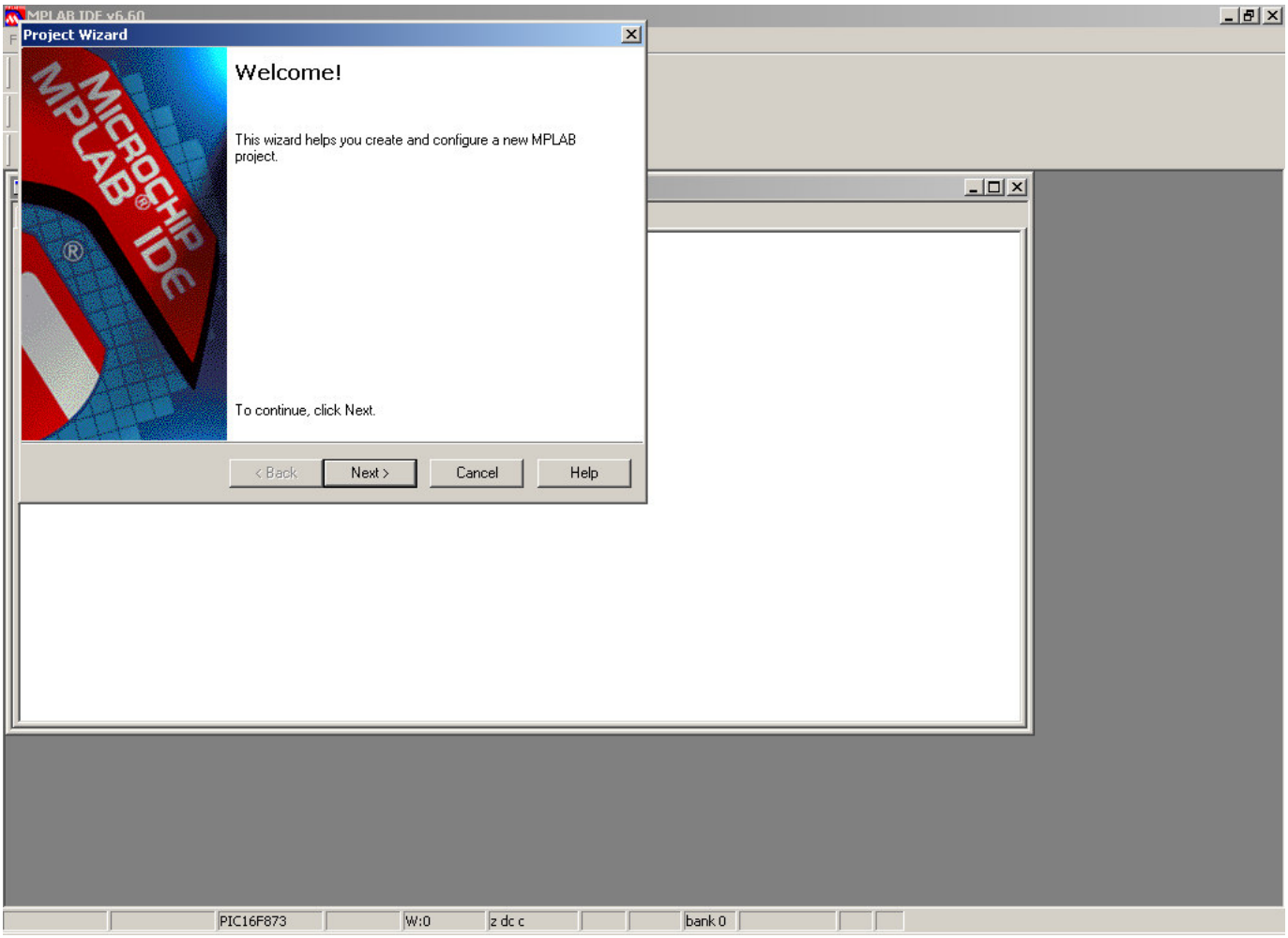
3.1. Abrindo um novo projeto para linguagem assembler.

Copie o arquivo comp_ass.asm para o diretório onde foi instalado o MPLAB.Caso você instalou no diretório recomando c:\arquivos de programas\MPLAB.

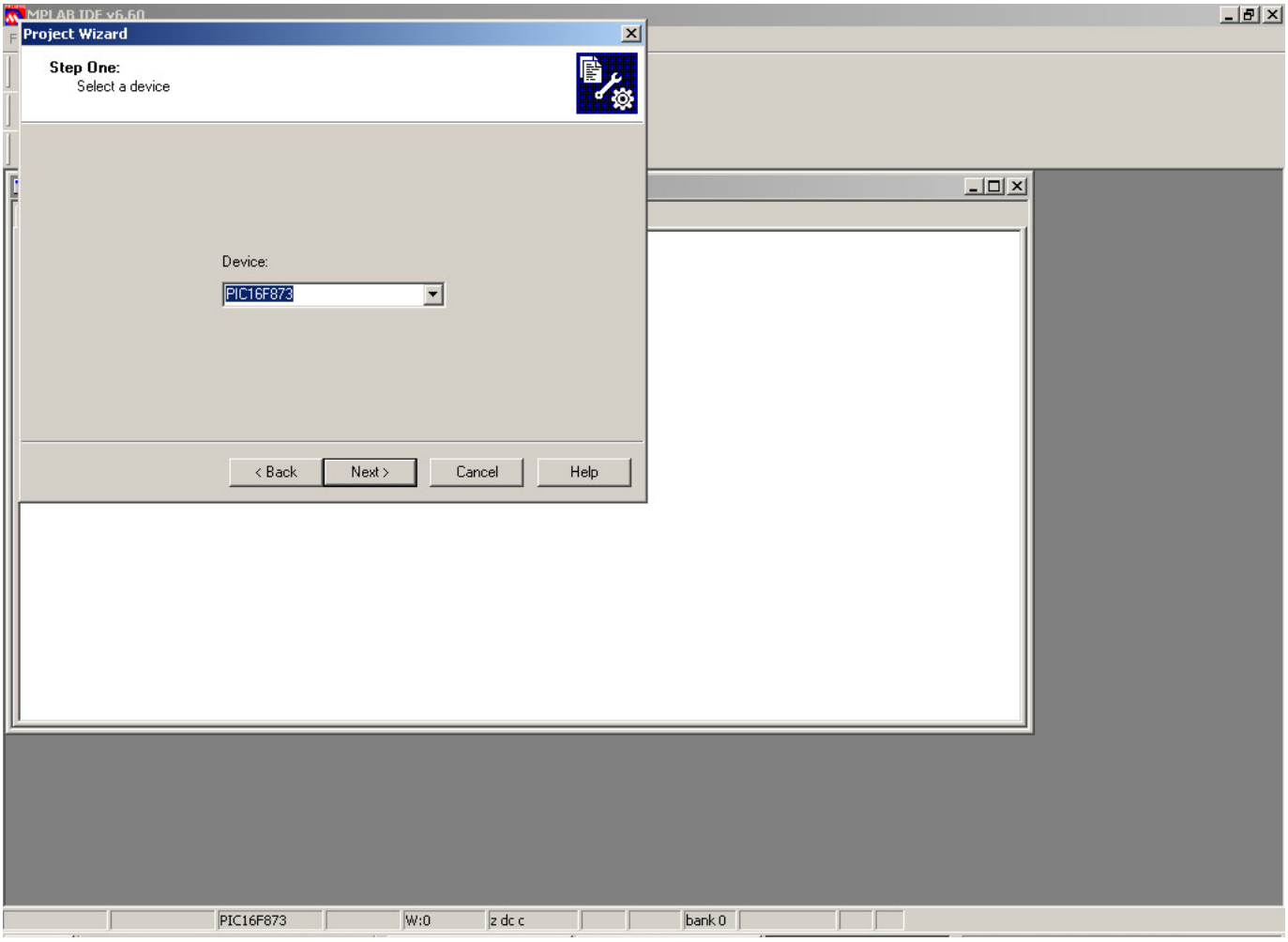
3.1.1 - Rode o programa MPLAB, clique no Project e sem seguida em New Project....



3.1.2 – Clique em Next.

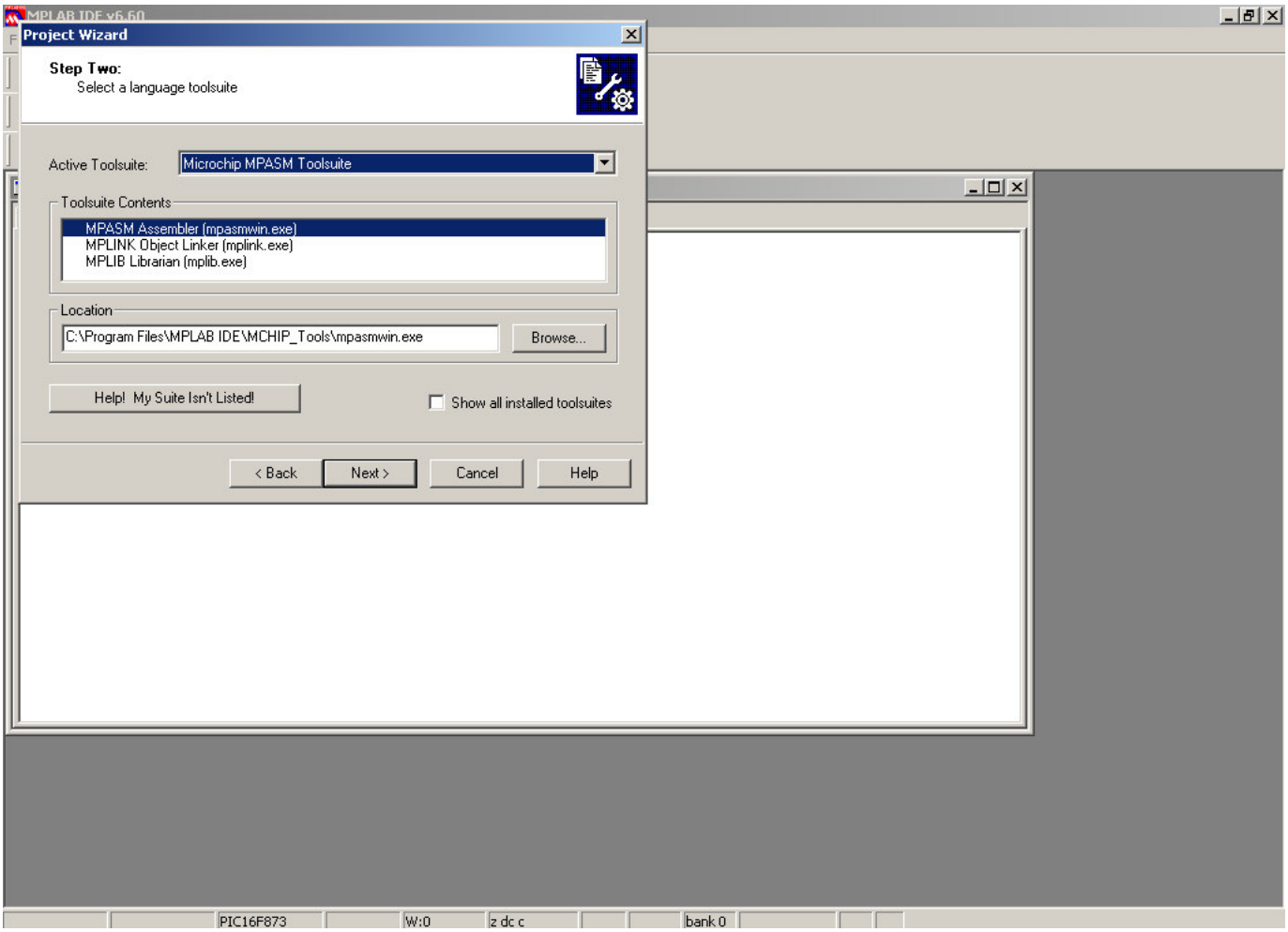


3.1.3 – Selecione o PIC correto para o seu projeto clique em Next.



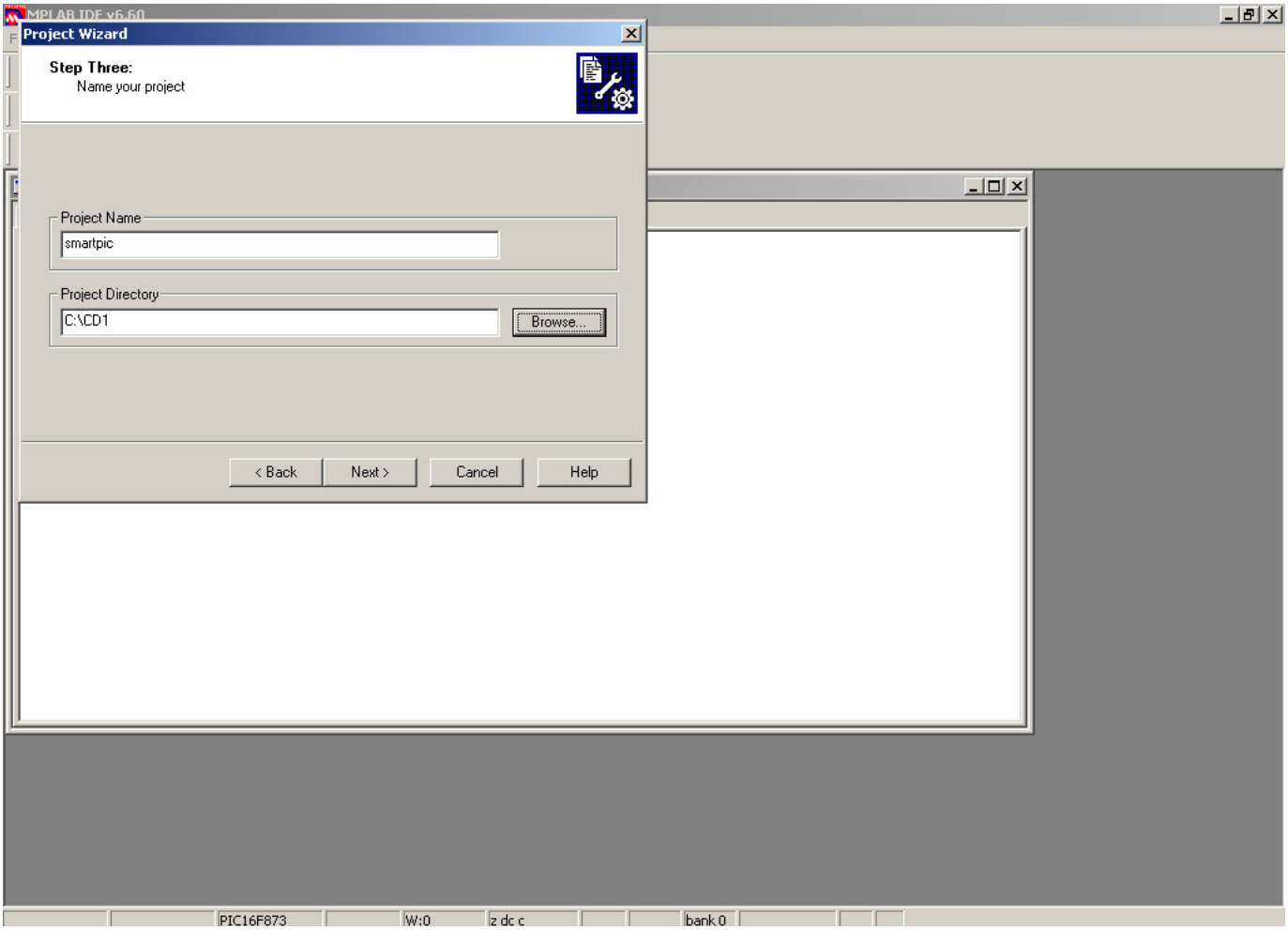
3.1.4 – Clique em Next.

Manual de Utilização do SMART PIC



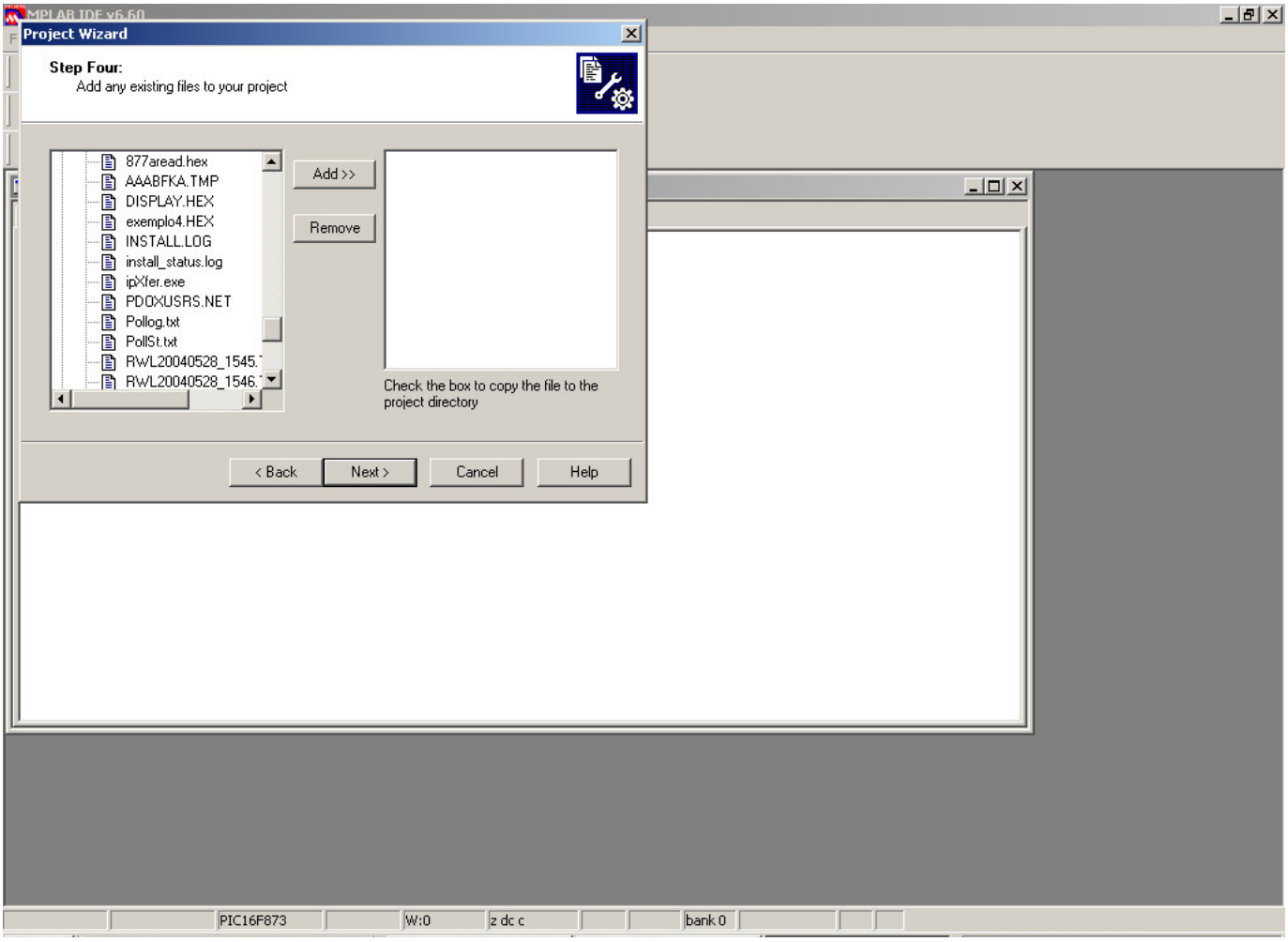
Manual de Utilização do SMART PIC

3.1.5 – Dê um nome ao seu projeto, selecione a pasta e clique em Next.

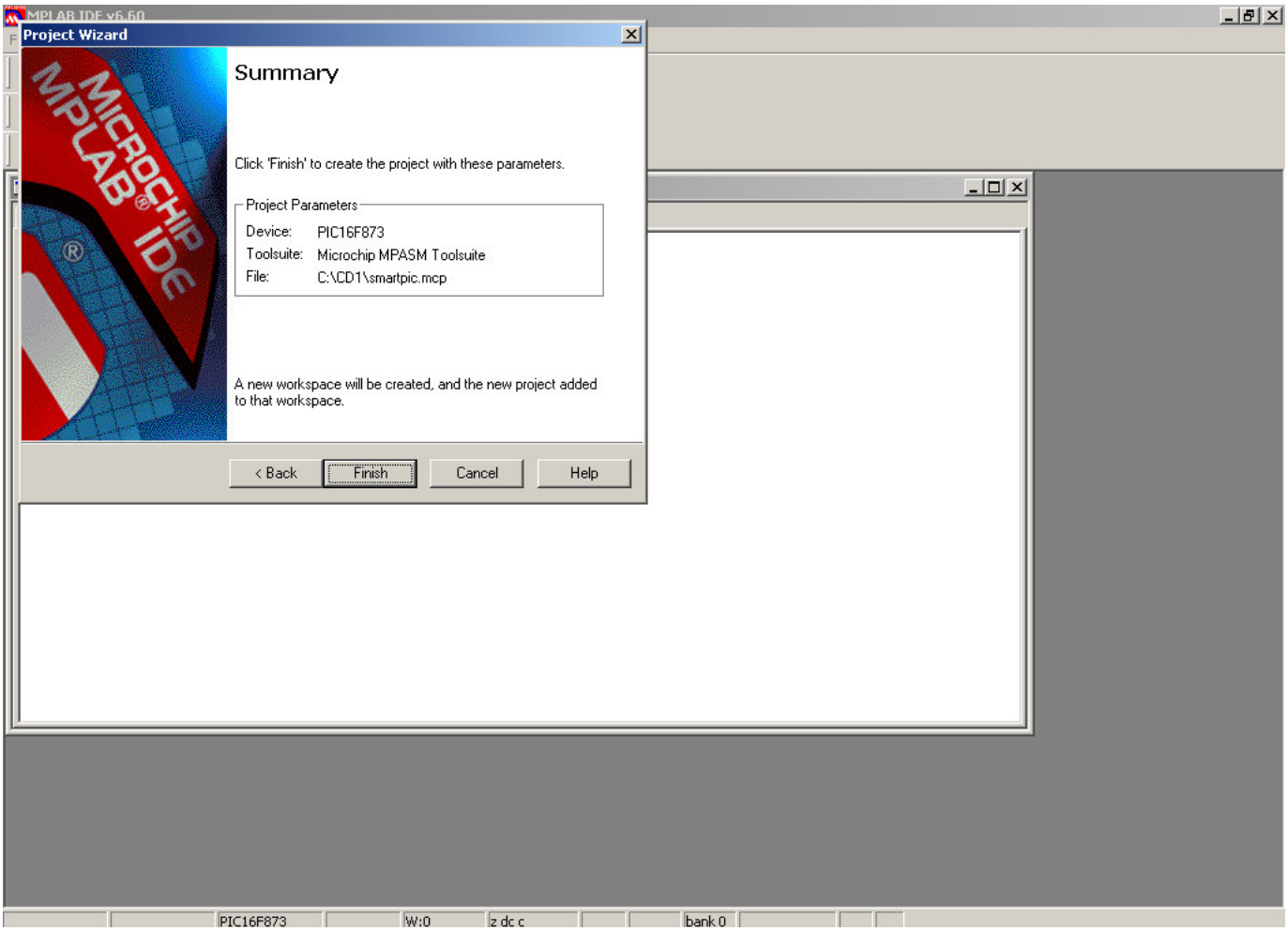


Manual de Utilização do SMART PIC

3.1.6 – Selecione um arquivo.asm somente se quiser usar um já existente senão clique em Next como no exemplo.



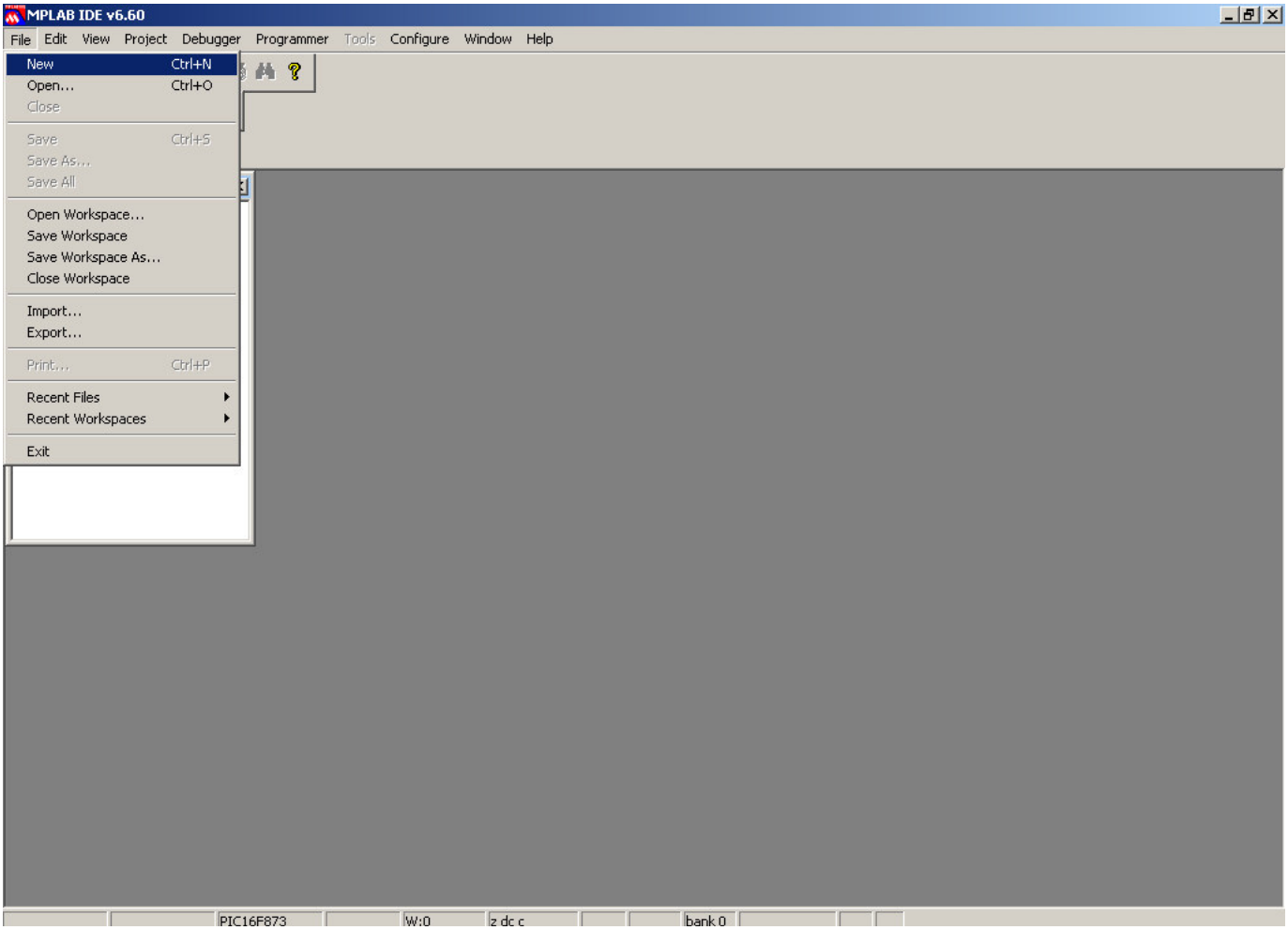
3.1.7 – Clique em Finish.



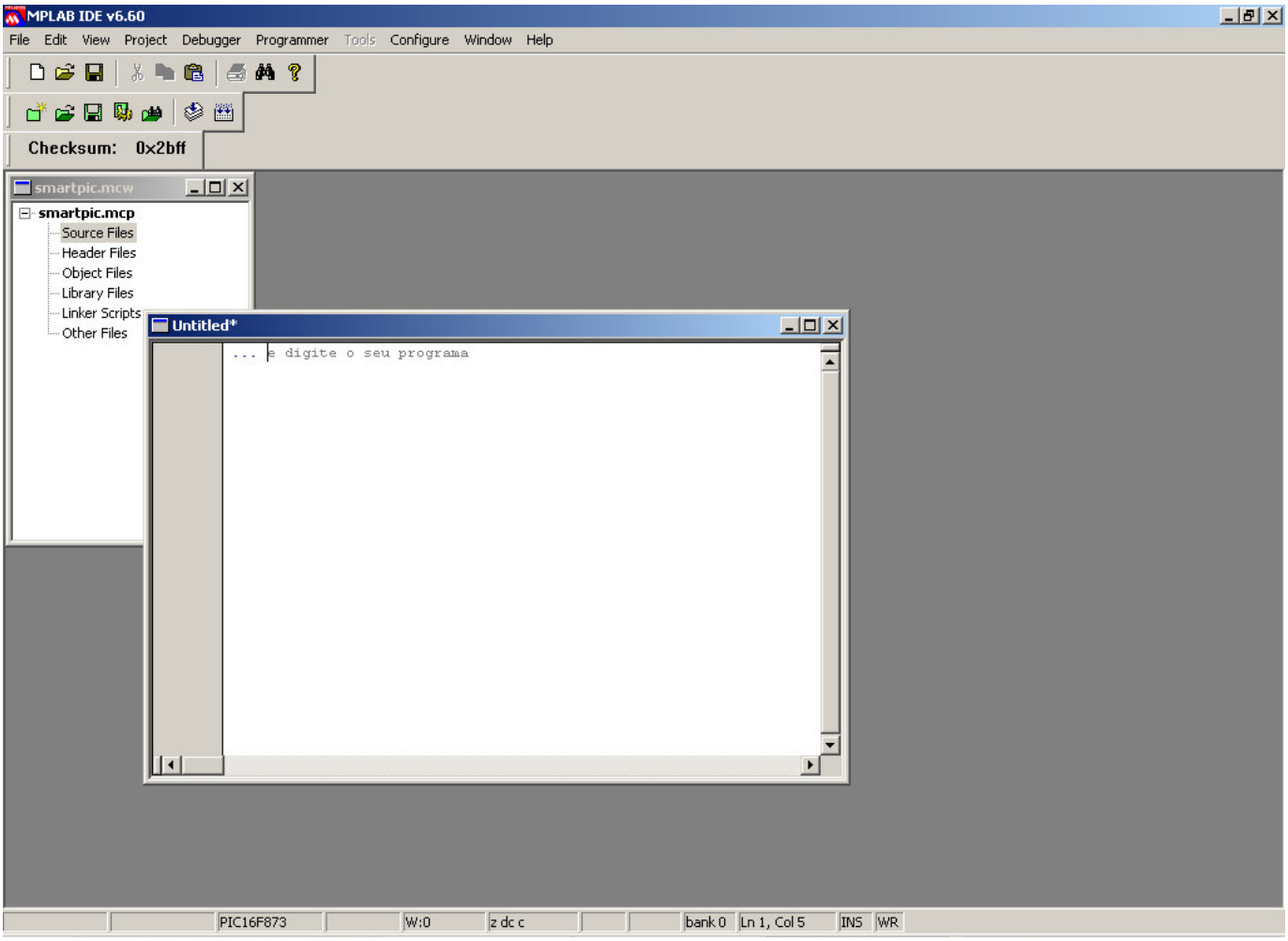
Manual de Utilização do SMART PIC

3.2 – Criando um Novo Programa.

Clique em File e em New.

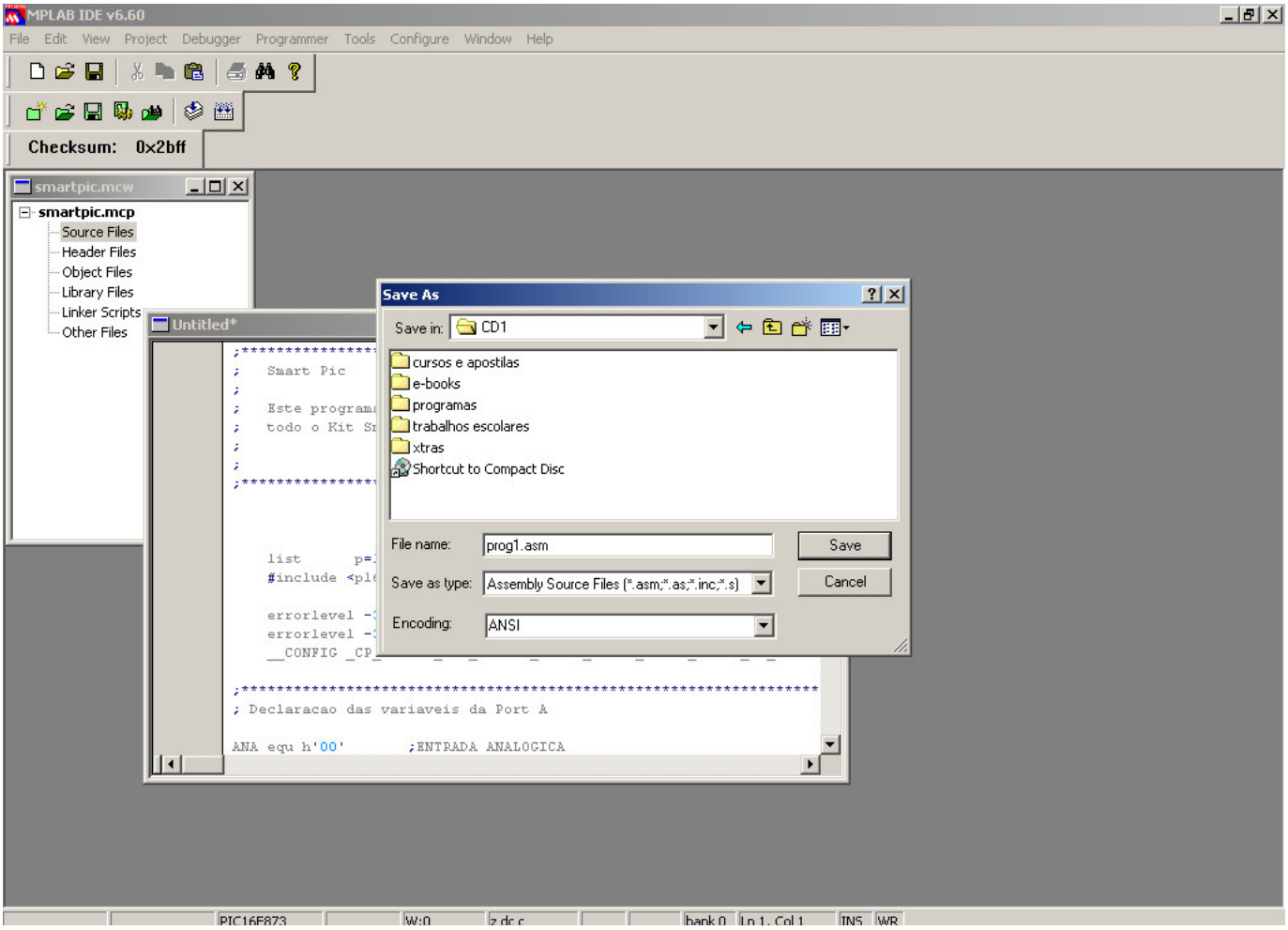


3.2.1– Digite aqui o seu programa.



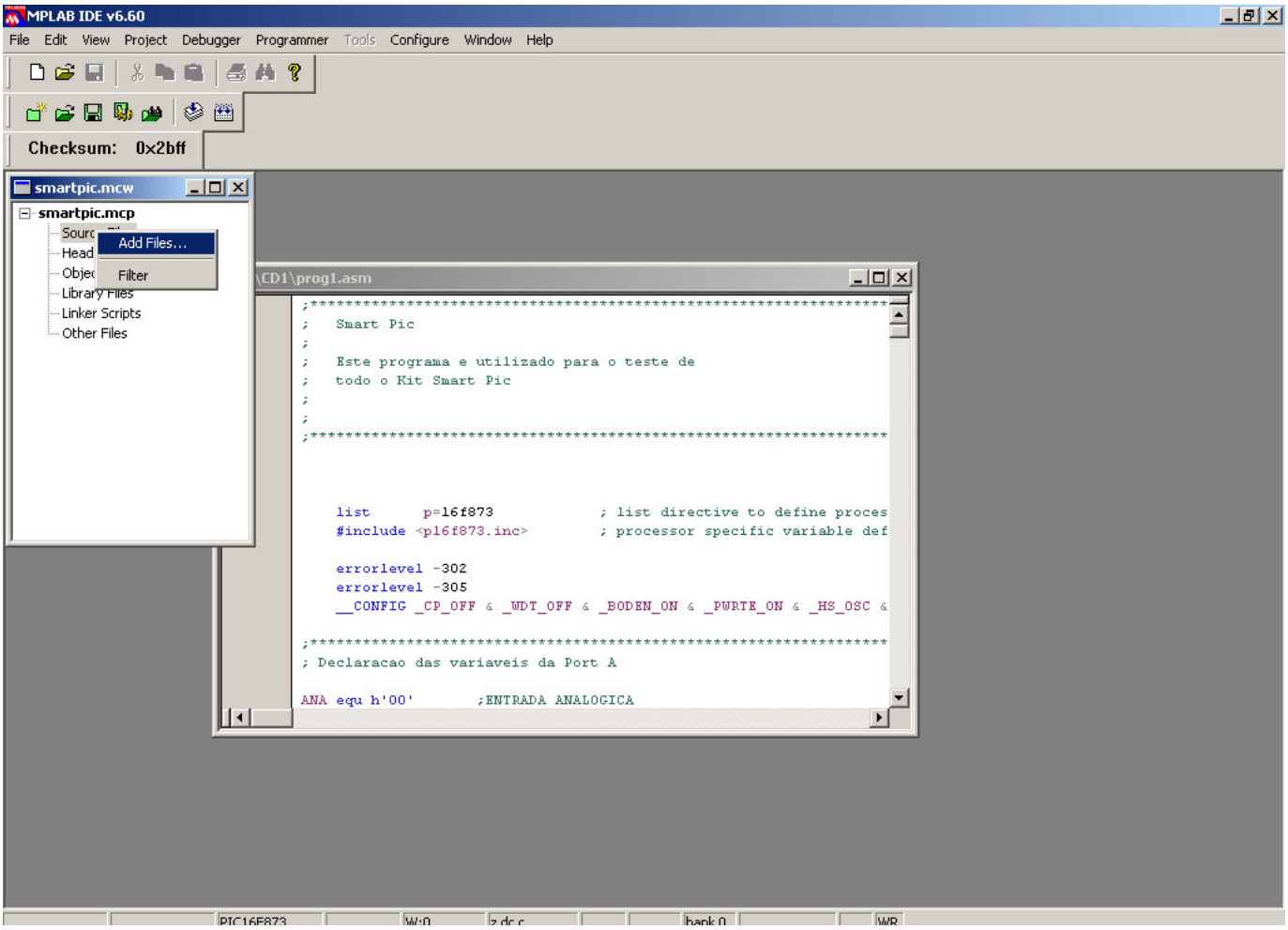
Manual de Utilização do SMART PIC

3.2.2 – Após digitar o programa selecione salvar como, selecione a pasta correta e dê um nome ao programa.



Manual de Utilização do SMART PIC

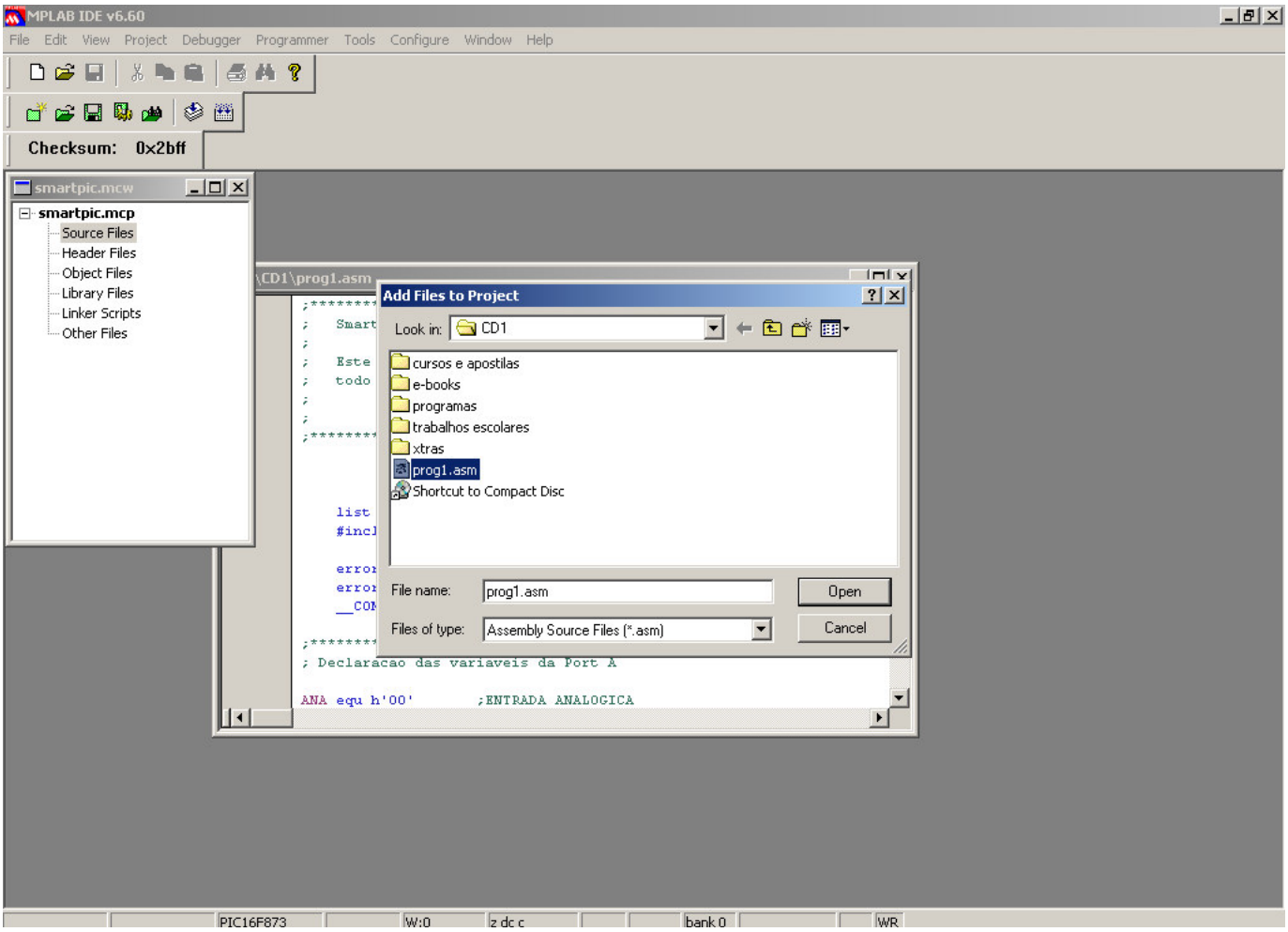
3.2.3 – Adicionando o programa ao seu projeto.
Clique com o botão direito no mouse em SORCE FILES e depois com o botão esquerdo clique em ADD FILES .


MPLAB IDE v6.60
File Edit View Project Debugger Programmer Tools Configure Window Help
Checksum: 0x2bff
smartpic.mcw
smartpic.mcp
Add Files...
Filter
Linker Scripts
Other Files
\CD1\prog1.asm
; Smart Pic
; Este programa e utilizado para o teste de
; todo o Kit Smart Pic
list p=16f873 ; list directive to define proces
#include <p16f873.inc> ; processor specific variable def
errorlevel -302
errorlevel -305
__CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON & _PWRTE_ON & _HS_OSC &
; Declaracao das variaveis da Port A
ANA equ h'00' ;ENTRADA ANALOGICA

Manual de Utilização do SMART PIC

3.2.4 – Adicionando um programa ao projeto.

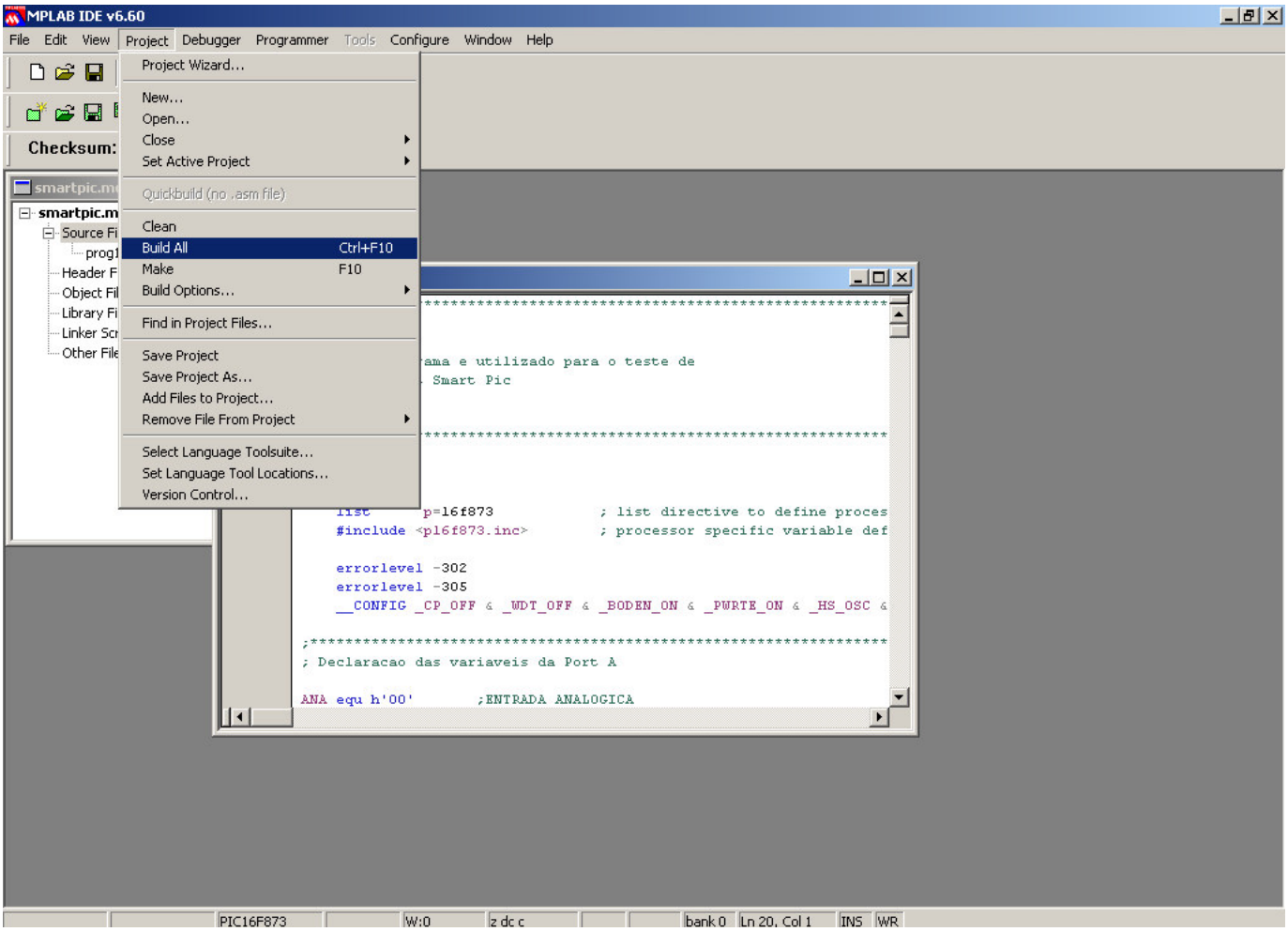
Selecione o arquivo e clique em Open.



Manual de Utilização do SMART PIC

3.3 – Compilando o programa.

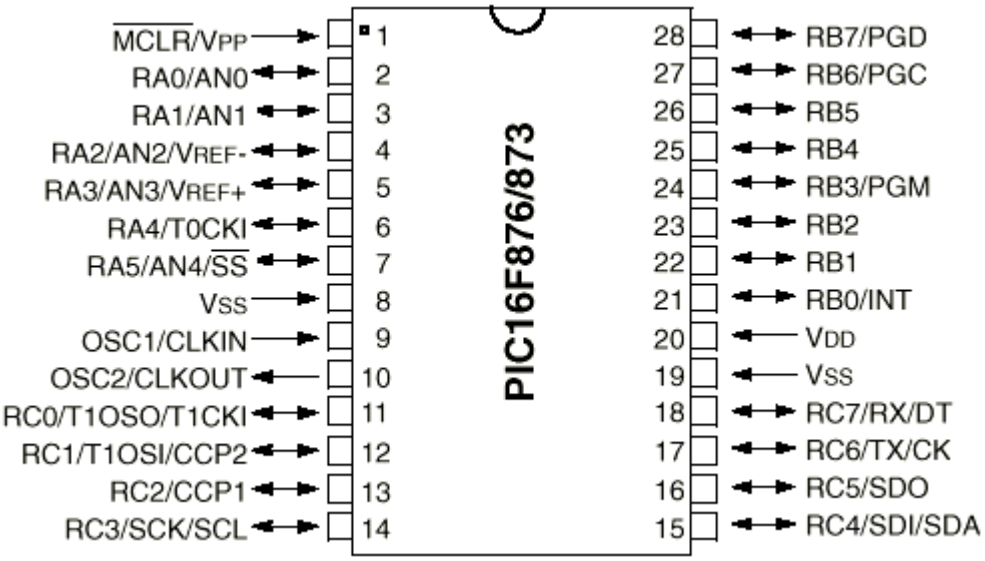
Após a conclusão do programa clique em PROJECT e em BUILD ALL para compilar o programa.



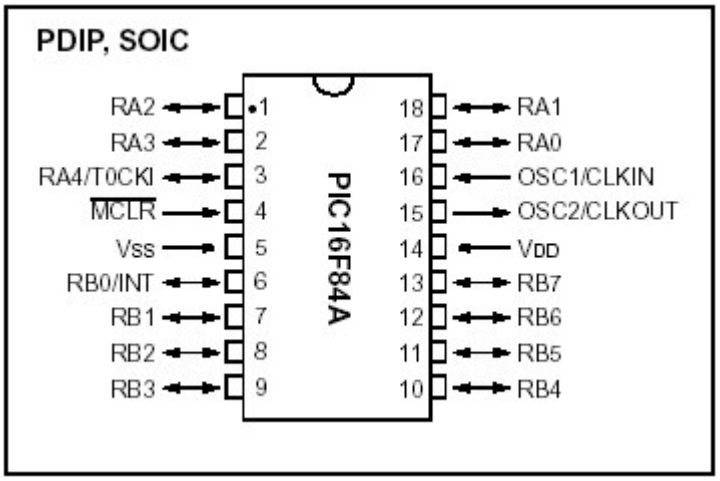
Manual de Utilização do SMART PIC

Capítulo 4

Pinagem do 16F873/A



Pinagem do 16F84/A



Manual de Utilização do SMART PIC

Capítulo 5

5. Conhecendo os mneumônicos do PIC – Apenas 35 instruções para aprender

Vamos começar apresentando uma tabela de acesso rápido as instruções do PIC, nas próximas páginas iremos detalhar uma a uma.

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 1fff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 1fff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

ADDLW — **Add Literal and W**

Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Encoding:	11 | 111x | kkkk | kkkk
Description:	The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.
Words:	1
Cycles:	1
Example	`ADDLW 0x15` Before Instruction W = 0x10 After Instruction W = 0x25

Esta instrução adiciona (soma) uma literal ao registrador W, o valor em W antes da instrução é perdido, pois o resultado é sobrescrito em W.
Vamos supor que o valor de W no momento seja 0x10.

Exemplo:
Antes da instrução
W=0x10
ADDLW 0x15
Após a instrução
W=0x25

Nota: a nomenclatura "0x15" indica que o valor será fornecido em hexadecimal ou também poderia ser escrito 15H. Para fornecer o valor em decimal coloque um ponto na frente. Exemplo .15.
O valor 0x15 é diferente do .15.

ADDWF — **Add W and f**

Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (dest)
Status Affected:	C, DC, Z
Encoding:	00 | 0111 | dfff | ffff
Description:	Add the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	`ADDWF FSR, 0` Before Instruction W = 0x17 FSR = 0xC2 After Instruction W = 0xD9 FSR = 0xC2

Esta instrução adiciona (soma) o conteúdo do registrador W com o registrador ´f´. Se ´d´ for 0, o resultado é enviado para W. Se ´d´for 1, o resultado é enviado para ´f ´.
Vamos supor que o valor de W no momento seja 0x17 e a variável CONTA declarada anteriormente no programa seja 0xC2.

Exemplo com ´d´ =0:
Antes da instrução
W=0x17, CONTA=0xC2
ADDWF CONTA ,0 <= aqui está o ´d´
Após a instrução
W=0xD9, CONTA=C2

Exemplo com ´d´ =1:
Antes da instrução
W=0x17, CONTA=0xC2
ADDWF CONTA ,1 <= aqui está o ´d´
Após a instrução
W=0x17, CONTA=D9

Manual de Utilização do SMART PIC

ANDLW — **And Literal with W**

Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Encoding:	11 | 1001 | kkkk | kkkk
Description:	The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1
Example	ANDLW 0x5F Before Instruction W = 0xA3 After Instruction W = 0x03

Esta instrução efetua a operação lógica AND (E) entre uma literal o valor no registrador W.
Vamos supor que o valor de W no momento seja 0xA3

Antes da instrução
W=0xA3

ANDLW 0x5F
Após a instrução
W=0x03

Vamos fazer a operação em binário
W =A3 = 10100011
5F = 01011111 AND

W =03 = 00000011

ANDWF — **AND W with f**

Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (dest)
Status Affected:	Z
Encoding:	00 | 0101 | dfff | ffff
Description:	AND the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	ANDWF FSR, 1 Before Instruction W = 0x17 FSR = 0xC2 After Instruction W = 0x17 FSR = 0x02

Esta instrução efetua a operação lógica AND (E) entre uma variável (registrador ´f´) e o registrador W. Se ´d´ for 0, o resultado é enviado para W. Se ´d´for 1, o resultado é enviado para ´f ´.
Vamos supor que o valor de W no momento seja 0x17 e a variável CONTA declarada anteriormente no programa seja 0xC2.
Exemplo com ´d´ =0:
Antes da instrução
W=0x17 , CONTA=0xC2

ANDWF CONTA ,0 <= aqui está o ´d´

Após a instrução
W=0x02, CONTA=C2
Exemplo com ´d´ =1:
Antes da instrução
W=0x17, CONTA=0xC2

ADDWF CONTA ,1 <= aqui está o ´d´

Após a instrução
W=0x17, CONTA=02
Vamos fazer a operação em binário
W =17 = 00010111
C2 = 11000010 AND

W =02 = 00000010

Manual de Utilização do SMART PIC

BSF	Bit Set f
Syntax:	[*label*] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Encoding:	01 | 01bb | bfff | ffff
Description:	Bit 'b' in register 'f' is set.
Words:	1
Cycles:	1
Example	BSF FLAG_REG, 7 Before Instruction FLAG_REG = 0x0A After Instruction FLAG_REG = 0x8A

Esta instrução é uma das mais importantes, pois ela pode atuar diretamente nos PORTS no PIC, portanto ela pode ligar ou desligar algo que esteja ligado no pino do PIC.
Esta instrução pode ainda atuar em uma variável declarada anteriormente no programa CONTA, supondo CONTA 0x08.
Exemplo ligando um led <= vamos supor que temos ligado um led no PORTC pino 18 (RC7), vamos ligar este led.

Antes da instrução
PORTC = 0000 0000
BSF PORTC,7
Depois da instrução
PORTC = 1000 0000

Exemplo setando um bit de uma variável CONTA.

Antes da instrução
CONTA = 0000 1010 = 0x0A
BSF CONTA,7
Depois da instrução
CONTA = 1000 1010 = 0x8A

BCF	Bit Clear f
Syntax:	[*label*] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → (f<b>)
Status Affected:	None
Encoding:	01 | 00bb | bfff | ffff
Description:	Bit 'b' in register 'f' is cleared.
Words:	1
Cycles:	1
Example	BCF FLAG_REG, 7 Before Instruction FLAG_REG = 0xC7 After Instruction FLAG_REG = 0x47

Esta instrução faz exatamente o oposto da instrução anterior, muito importante também, pois ela pode atuar diretamente nos PORTS no PIC, portanto ela pode ligar ou desligar algo que esteja ligado no pino do PIC.
Esta instrução pode ainda atuar em uma variável declarada anteriormente no programa CONTA, supondo CONTA 0x08.
Exemplo desligando um led <= vamos supor que temos ligado um led no PORTC pino 18 (RC7), vamos desligar este led.

Antes da instrução
PORTC = 1000 0000
BSF PORTC,7
Depois da instrução
PORTC = 0000 0000

Exemplo resetando um bit de uma variável CONTA.

Antes da instrução
CONTA = 1000 1010 = 0x8A
BSF CONTA,7
Depois da instrução
CONTA = 0000 1010 = 0x0A

Manual de Utilização do SMART PIC

BTFSC — **Bit Test, Skip if Clear**

Syntax: [*label*] BTFSC f,b

Operands: 0 ≤ f ≤ 127
0 ≤ b ≤ 7

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

01	10bb	bfff	ffff

Description: If bit 'b' in register 'f' is '0', then the next instruction is skipped.
If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded, and a NOP is executed instead, making this a 2 cycle instruction.

Words: 1

Cycles: 1(2)

Example

```
HERE    BTFSC   FLAG,1
FALSE   GOTO    PROCESS_CO
TRUE    •       DE
        •
        •
```

Before Instruction
PC = address HERE
After Instruction
if FLAG<1> = 0,
PC = address TRUE
if FLAG<1>=1,
PC = address FALSE

BTFSC, Testa um Bit de um PORT ou de uma variável e pula a instrução imediatamente abaixo caso o bit esteja em 0, muito utilizada para testar se um botão esta apertado ou não.

Exemplo testando um botão <= vamos supor que temos um botão ligado no PORTA pino 3 (RA1), vamos testar se o botão esta apertado, neste exemplo botão apertado = 5v e solto= 0v.

```
BTFSC     PORTA,1
Goto      botão_apertado
Goto      botão_solto
```

Exemplo testando um bit de uma variável CONTA.

```
BTFSC     CONTA,1
Goto      bit_1_da_variável_conta_esta_em_1
Goto      bit_1_da_variável_conta_esta_em_0
```

BTFSC pula a instrução imediatamente abaixo, portanto se o PORTA bit 1 (pino 3) estiver em 0, o programa irá pular a instrução Goto Botão_apertado e irá diretamente para a instrução Goto Botão_solto.
BTFSC pula a instrução imediatamente abaixo, portanto se a variável CONTA bit 1 estiver em 0, o programa irá pular a instrução Goto bit_1_da_variável_conta_esta_em_1 e irá diretamente para a instrução Goto bit_1_da_variável_conta_esta_em_0.

Manual de Utilização do SMART PIC

BTFSS **Bit Test f, Skip if Set**

Syntax: [*label*] BTFSS f,b

Operands: 0 ≤ f ≤ 127
0 ≤ b < 7

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

01	11bb	bfff	ffff

Description: If bit 'b' in register 'f' is '1', then the next instruction is skipped.
If bit 'b' is '1', then the next instruction fetched during the current instruction execution, is discarded and a NOP is executed instead, making this a 2 cycle instruction.

Words: 1

Cycles: 1(2)

Example

```
HERE    BTFSS   FLAG,1
FALSE   GOTO    PROCESS_CO
TRUE    •       DE
        •
        •
```

Before Instruction
PC = address HERE
After Instruction
if FLAG<1> = 0,
PC = address FALSE
if FLAG<1> = 1,
PC = address TRUE

BTFSS, Testa um Bit de um PORT ou de uma variável e pula a instrução imediatamente abaixo caso o bit esteja em 1, muito utilizada para testar se um botão esta apertado ou não.

Exemplo testando um botão <= vamos supor que temos um botão ligado no PORTA pino 3 (RA1), vamos testar se o botão esta apertado, neste exemplo botão apertado = 5v e solto= 0v..

```
BTFSS     PORTA,1
Goto      botão_solto
Goto      botão_apertado
```

Exemplo testando um bit de uma variável CONTA.

```
BTFSS     CONTA,1
Goto      bit_1_da_variável_conta_esta_em_0
Goto      bit_1_da_variável_conta_esta_em_1
```

BTFSS pula a instrução imediatamente abaixo, portanto se o PORTA bit 1 (pino 3) estiver em 1, o programa irá pular a instrução Goto Botão_solto e irá diretamente para a instrução Goto Botão_apertado.
BTFSS pula a instrução imediatamente abaixo, portanto se a variável CONTA bit 1 estiver em 0, o programa irá pular a instrução Goto bit_1_da_variável_conta_esta_em_0 e irá diretamente para a instrução Goto bit_1_da_variável_conta_esta_em_1.

Manual de Utilização do SMART PIC

CALL — **Call Subroutine**

Syntax:	[*label*] CALL k
Operands:	0 ≤ k ≤ 2047
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Affected:	None
Encoding:	10 | 0kkk | kkkk | kkkk
Description:	Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two cycle instruction.
Words:	1
Cycles:	2
Example	HERE CALL THERE Before Instruction PC = Address HERE After Instruction PC = Address THERE TOS = Address HERE+1

Esta instrução efetua chamada de subrotina

Ex: O programa é desviado para o endereço indicado e é somado 1 ao Apontador para o retorno da subrotina.

Call LER_EEPROM ; supondo que esteja no endereço 19H então será salvo no Apontador 20H que será a linha de retorno da subrotina.

LER_EEPROM ; Subrotina chamada
Return ; retorno da subrotina.

OBS: O retorno de subrotina será estudado adiante.

CLRF — **Clear f**

Syntax:	[*label*] CLRF f
Operands:	0 ≤ f ≤ 127
Operation:	00h → (f) 1 → Z
Status Affected:	Z
Encoding:	00 | 0001 | 1fff | ffff
Description:	The contents of register 'f' are cleared and the Z bit is set.
Words:	1
Cycles:	1
Example	CLRF FLAG_REG Before Instruction FLAG_REG = 0x5A After Instruction FLAG_REG = 0x00 Z = 1

Esta instrução zera o registro f.

Ex : Temos um ´f ´declarado como, VALOR

Antes da instrução
VALOR=FAH

CLRF VALOR

Depois da instrução VALOR ficará igual a Zero e o bit Z de STATUS, ficará igual a 1.

CLRW — **Clear W**

Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Encoding:	00 | 0001 | 0000 | 0011
Description:	W register is cleared. Zero bit (Z) is set.
Words:	1
Cycles:	1
Example	`CLRW` Before Instruction W = 0x5A After Instruction W = 0x00 Z = 1

Esta instrução zera o registrador W.

Ex : Temos com exemplo antes W estava igual a 02H

Antes da instrução
W=02H

CLRW

Depois da instrução W ficará igual a Zero e o bit Z de STATUS, ficará igual a 1.

CLRWDT — **Clear Watchdog Timer**

Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h → WDT 0 → WDT prescaler, 1 → $\overline{TO}$ 1 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Encoding:	00 | 0000 | 0110 | 0100
Description:	`CLRWDT` instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.
Words:	1
Cycles:	1
Example	`CLRWDT` Before Instruction WDT counter = ? After Instruction WDT counter = 0x00 WDT prescaler = 0 $\overline{TO}$ = 1 $\overline{PD}$ = 1

Esta instrução reseta o Watch Dog.

Ex : O registro de Watch Dog será zerado, assim como o registro do prescaler (se estiver direcionado ao Watch Dog). BITS TO\ e PD\ são setados.

CLRWDT

Após a instrução o registro de Watch Dog que estava contando volta a zero 0, e se estiver usando prescaler, seu registro também será zerado.

Importante: Se o Watch Dog estiver habilitado, esta instrução deverá ser executada de tempos em tempos pelo programa, para evitar o Time_Out do registro do Watch Dog e conseqüentemente o reset do sistema.

Manual de Utilização do SMART PIC

COMF — **Complement f**

Syntax:	[*label*] COMF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	$(\bar{f}) \rightarrow (dest)$
Status Affected:	Z
Encoding:	00 | 1001 | dfff | ffff
Description:	The contents of register 'f' are complemented. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	`COMF REG1,0` Before Instruction REG1 = 0x13 After Instruction REG1 = 0x13 W = 0xEC

Esta instrução Inverte os bits de "f".

Ex : Todos os bits de "f" um a um serão invertidos, O que era 0 ficará 1 e vice versa.
CLRWDT

Após a instrução o registro de Watch Dog que estava contando volta a zero 0, e se estiver usando prescaler, seu registro também será zerado.

Importante: Se o Watch Dog estiver habilitado, esta instrução deverá ser executada de tempos em tempos pelo programa, para evitar o Time_Out do registro do Watch Dog e conseqüentemente o reset do sistema.

DECF — **Decrement f**

Syntax:	[*label*] DECF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (dest)
Status Affected:	Z
Encoding:	00 | 0011 | dfff | ffff
Description:	Decrement register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	`DECF CNT, 1` Before Instruction CNT = 0x01 Z = 0 After Instruction CNT = 0x00 Z = 1

Esta instrução diminui em 1 a quantidade armazenada em f.

Ex : Temos com exemplo antes f estava igual a 04H
Se d= 1 o resultado será salvo em f.
Se d= 0 o resultado será salvo em W.

Antes da instrução

f= VALOR

VALOR=04H

decf VALOR,1

Depois da instrução "f" ficará igual a 03H.

Manual de Utilização do SMART PIC

DECFSZ — **Decrement f, Skip if 0**

Syntax: [*label*] DECFSZ f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - 1 → (dest); skip if result = 0

Status Affected: None

Encoding:

00	1011	dfff	ffff

Description: The contents of register 'f' are decremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.
If the result is 0, the next instruction, which is already fetched, is discarded. A NOP is executed instead making it a two cycle instruction.

Words: 1

Cycles: 1(2)

Example

```
HERE      DECFSZ   CNT, 1
          GOTO     LOOP
CONTINUE  •
          •
          •
```

Before Instruction
PC = address HERE
After Instruction
CNT = CNT - 1
if CNT = 0,
PC = address CONTINUE
if CNT ≠ 0,
PC = address HERE+1

Esta instrução diminui em 1 a quantidade armazenada em f. E se f = 0 pula a próxima instrução se não executa a próxima instrução.

Ex : Temos com exemplo antes f estava igual a 01H
Se d= 1 o resultado será salvo em f.
Se d= 0 o resultado será salvo em W.

Antes da instrução

f= VALOR

VALOR=01H

```
decfsz      VALOR,1  ; valor=valor-1
goto        Mais     ; como valor=0 pula
goto        Menos    ; executa esta instrução
```

Se usarmos decfsz VALOR,0 a operação será W= VALOR – 1 e o resultado será salvo em W.

Manual de Utilização do SMART PIC

GOTO	Unconditional Branch
Syntax:	[*label*] GOTO k
Operands:	0 ≤ k ≤ 2047
Operation:	k → PC<10:0> PCLATH<4:3> → PC<12:11>
Status Affected:	None
Encoding:	10 | 1kkk | kkkk | kkkk
Description:	GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two cycle instruction.
Words:	1
Cycles:	2
Example	GOTO THERE After Instruction PC = Address THERE

Esta instrução desvia para o endereço K.

Este é um desvio incondicional indicado por K ou seja não depende de nenhuma condição.

Ex : Temos o trecho de programa abaixo

```
Inicio
        MOVLW       10H     ; move 10h para W
        GOTO        Limpa   ; desvia para Limpa
.........
.........
Limpa
        CLRW                ; limpa W
        GOTO        Inicio  ; desvia para Inicio
```

INCF	Increment f
Syntax:	[*label*] INCF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (dest)
Status Affected:	Z
Encoding:	00 | 1010 | dfff | ffff
Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.
Words:	1
Cycles:	1
Example	INCF CNT, 1 Before Instruction CNT = 0xFF Z = 0 After Instruction CNT = 0x00 Z = 1

Esta instrução soma 1 a quantidade armazenada em f.

Ex : "f"=('f'+1)
Temos com exemplo antes f estava igual a 04H
Se d= 1 o resultado será salvo em f.
Se d= 0 o resultado será salvo em W.

Antes da instrução

f= VALOR

VALOR=04H

```
INCF        VALOR,1
```

Depois da instrução "f" ficará igual a 05H.

Manual de Utilização do SMART PIC

INCFSZ — **Increment f, Skip if 0**

Syntax: [*label*] INCFSZ f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) + 1 → (dest), skip if result = 0

Status Affected: None

Encoding:

00	1111	dfff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.
If the result is 0, the next instruction, which is already fetched, is discarded. A NOP is executed instead making it a two cycle instruction.

Words: 1

Cycles: 1(2)

Example

```
HERE      INCFSZ    CNT,  1
          GOTO      LOOP
CONTINUE  •
          •
          •
```

Before Instruction
PC = address HERE
After Instruction
CNT = CNT + 1
if CNT= 0,
PC = address CONTINUE
if CNT≠ 0,
PC = address HERE +1

Esta instrução soma 1 a quantidade armazenada em f. E se f = 0 pula a próxima instrução se não executa a próxima instrução.

Ex : Temos com exemplo antes f estava igual a FFH
Se d= 1 o resultado será salvo em f.
Se d= 0 o resultado será salvo em W.

Antes da instrução

f= VALOR

VALOR=FFH

```
incfsz      VALOR,1  ; valor=valor-1
goto        Mais     ; como valor=0 pula
goto        Menos    ; executa esta instrução
```

Se usarmos decfsz VALOR,0 a operação será W= VALOR + 1 e o resultado será salvo em W.

IORLW **Inclusive OR Literal with W**

Syntax:	[*label*] IORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .OR. k → (W)
Status Affected:	Z
Encoding:	11 | 1000 | kkkk | kkkk
Description:	The contents of the W register are OR'ed with the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1
Example	IORLW 0x35 Before Instruction W = 0x9A After Instruction W = 0xBF Z = 1

Esta instrução realiza a operação OR entre o conteúdo de W e uma constante K.

Ex : Temos W=96H
A intrução IORLW 17H resultará em W=97H

W=96H= 100100110
OR 17H= 00010111
10010111

IORWF **Inclusive OR W with f**

Syntax:	[*label*] IORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .OR. (f) → (dest)
Status Affected:	Z
Encoding:	00 | 0100 | dfff | ffff
Description:	Inclusive OR the W register with register 'f'. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.
Words:	1
Cycles:	1
Example	IORWF RESULT, 0 Before Instruction RESULT = 0x13 W = 0x91 After Instruction RESULT = 0x13 W = 0x93 Z = 1

Esta instrução realiza a operação OR entre o conteúdo de W e o conteúdo de "f" .

Se d= 1 o resultado será salvo em f.
Se d= 0 o resultado será salvo em W.

Ex : Temos W=96H
VALOR=16H
A intrução IORLW VALOR,1 resultará em W=9EH

W=96H= 100100110
OR 18H= 00011000
10011110

MOVLW **Move Literal to W**

Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W)
Status Affected:	None
Encoding:	11 | 00xx | kkkk | kkkk
Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.
Words:	1
Cycles:	1
Example	MOVLW 0x5A After Instruction W = 0x5A

Esta instrução move o conteúdo da constante K para W.

Ex : Temos 16H como constante.

MOVLW 16H

Resultará em W=16H

MOVF **Move f**

Syntax:	[*label*] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (dest)
Status Affected:	Z
Encoding:	00 | 1000 | dfff | ffff
Description:	The contents of register f are moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.
Words:	1
Cycles:	1
Example	MOVF FSR, 0 After Instruction W = value in FSR register Z = 1

Esta instrução move o conteúdo de "f" para o destino d.

Se d= 1 o resultado será salvo em f.
Se d= 0 o resultado será salvo em W.

Ex : Temos W=96H
VALOR=16H

MOVF VALOR,0

Resultará em W=16H

MOVWF **Move W to f**

Syntax:	[*label*] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Encoding:	00 | 0000 | 1fff | ffff
Description:	Move data from W register to register 'f'.
Words:	1
Cycles:	1
Example	MOVWF OPTION Before Instruction OPTION = 0xFF W = 0x4F After Instruction OPTION = 0x4F W = 0x4F

Esta instrução move o conteúdo de W para "f". "f"=W

Ex : Temos 16H em W.

```
MOVWF       VALOR
```

Resultará em VALOR=16H

NOP **No Operation**

Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	00 | 0000 | 0xx0 | 0000
Description:	No operation.
Words:	1
Cycles:	1
Example	NOP

Esta instrução é de nenhuma operação, apenas perde um ciclo de máquina.

Ex :

```
MOVLW       16H         ; move 34h para W
NOP                     ; perde 1 ciclo.
MOVF        VALOR,0     ; W = VALOR
```

RETFIE	Return from Interrupt
Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None
Encoding:	00 | 0000 | 0000 | 1001
Description:	Return from Interrupt. Stack is POPed and Top of Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a two cycle instruction.
Words:	1
Cycles:	2
Example	`RETFIE` After Interrupt PC = TOS GIE = 1

Esta instrução é usada para fazer um retorno de interrupção. O endereço de retorno é recuperado no STACK e o bit GIE é setado.

Ex :

```
MOVLW   16H     ; move 34h para W
NOP             ; perde 1 ciclo.
RETFIE          ; Retorno de Interrupção
```

Manual de Utilização do SMART PIC

RETLW — **Return with Literal in W**

Syntax: [*label*] RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → (W);
TOS → PC

Status Affected: None

Encoding:

11	01xx	kkkk	kkkk

Description: The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two cycle instruction.

Words: 1

Cycles: 2

Example

```
CALL TABLE;W contains
table
        ;offset value
TABLE   •       ;W now has table value
        •
        •
ADDWF PC    ;W = offset
RETLW k1    ;Begin table
RETLW k2    ;
        •
        •
        •
RETLW kn    ; End of table

Before Instruction
        W   =   0x07
After Instruction
        W   =   value of k8
```

Esta instrução é usada para retorno de subrotina com W=K (constante).
O endereço de retorno é recuperado no STACK.

Ex :

```
MOVLW   16H     ; move 16H para W
NOP             ; perde 1 ciclo.
RETLW   01H     ; W = 1 e retorna
```

Manual de Utilização do SMART PIC

RETURN — **Return from Subroutine**

Syntax: [*label*] RETURN

Operands: None

Operation: TOS → PC

Status Affected: None

Encoding:

00	0000	0000	1000

Description: Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two cycle instruction.

Words: 1

Cycles: 2

Example

```
RETURN
```

After Interrupt
PC = TOS

Esta instrução é usada para retorno de subrotina.
O endereço de retorno é recuperado no STACK.

Ex :

```
MOVLW   16H     ; move 16H para W
NOP             ; perde 1 ciclo.
RETURN          ; retorna  da subrotina.
```

Importante Stack total de oito níveis.

RLF — **Rotate Left f through Carry**

Syntax: [*label*] RLF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

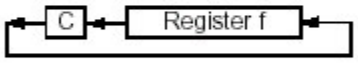
Operation: See description below

Status Affected: C

Encoding:

00	1101	dfff	ffff

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is stored back in register 'f'.



Words: 1

Cycles: 1

Example

```
RLF     REG1,0
```

Before Instruction
REG1 = 1110 0110
C = 0
After Instruction
REG1 = 1110 0110
W = 1100 1100
C = 1

Esta instrução rotaciona à esquerda pelo carry. O valor no registro de "f" é rotacionado em um bit para a esquerda, passando pelo carry.
OBS: Bit afetado C.
Se d= 1 o resultado será salvo em f.
Se d= 0 o resultado será salvo em W.
Ex :
VALOR=00110011 e C=1

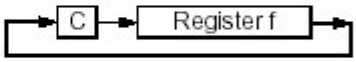
```
RLF         VALOR,1
```

Após a instrução
VALOR = 01100111 e C=0

Manual de Utilização do SMART PIC

RRF — **Rotate Right f through Carry**

Syntax:	[*label*] RRF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	See description below
Status Affected:	C
Encoding:	00 | 1100 | dfff | ffff
Description:	The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.
Words:	1
Cycles:	1
Example	RRF REG1, 0



Before Instruction
REG1 = 1110 0110
C = 0
After Instruction
REG1 = 1110 0110
W = 0111 0011
C = 0

Esta instrução rotaciona à direita pelo carry. O valor no registro de "f" é rotacionado em um bit para a direita, passando pelo carry.
OBS: Bit afetado C.
Se d= 1 o resultado será salvo em f.
Se d= 0 o resultado será salvo em W.
Ex :
VALOR=00110011 e C=1

RLF VALOR,1

Após a instrução
VALOR = 10011001 e C=1

SLEEP

Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h → WDT, 0 → WDT prescaler, 1 → $\overline{TO}$, 0 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Encoding:	00 | 0000 | 0110 | 0011
Description:	The power-down status bit, $\overline{PD}$ is cleared. Time-out status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.
Words:	1
Cycles:	1
Example:	SLEEP

Esta instrução SLEEP o oscilador para e o Watch Dog e o Prescaler são zerados.

Ex :

SLEEP

Esta instrução é muito utilizada quando se necessita de baixo consumo do sistema pois nessa condição o micro consume na ordem de micro ampéres.

Manual de Utilização do SMART PIC

SUBLW **Subtract W from Literal**

Syntax: [*label*] SUBLW k

Operands: 0 ≤ k ≤ 255

Operation: k - (W) → (W)

Status Affected: C, DC, Z

Encoding:

11	110x	kkkk	kkkk

Description: The W register is subtracted (2's complement method) from the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example 1:
```
SUBLW   0x02
```
Before Instruction

W = 1
C = ?

After Instruction

W = 1
C = 1; result is positive

Example 2: Before Instruction

W = 2
C = ?

After Instruction

W = 0
C = 1; result is zero

Example 3: Before Instruction

W = 3
C = ?

After Instruction

W = 0xFF
C = 0; result is negative

Esta instrução é utilizada quando se deseja Subtrair W de uma constante K.
W=K-W

Registros afetados: C, DC, Z.

Ex 1:
Temos W =1H
SUBLW 2H ; resultará W=2-1 = 1 C=1 (+)
Ex 2:
Temos W =3H
SUBLW 2H ; resultará W=2-3 = -1 C=0 (-)

Manual de Utilização do SMART PIC

SUBWF **Subtract W from f**

Syntax: [*label*] SUBWF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) - (W) → (dest)

Status Affected: C, DC, Z

Encoding:

00	0010	dfff	ffff

Description: Subtract (2's complement method) W register from register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example 1:
```
SUBWF   REG1,1
```
Before Instruction
```
REG1  =  3
W     =  2
C     =  ?
```
After Instruction
```
REG1  =  1
W     =  2
C     =  1; result is positive
```

Example 2: Before Instruction
```
REG1  =  2
W     =  2
C     =  ?
```
After Instruction
```
REG1  =  0
W     =  2
C     =  1; result is zero
```

Example 3: Before Instruction
```
REG1  =  1
W     =  2
C     =  ?
```
After Instruction
```
REG1  =  0xFF
W     =  2
C     =  0; result is negative
```

Esta instrução é utilizada quando se deseja Subtrair W de um regitro "f".
W = f - W

Registros afetados: C, DC, Z.

Ex 1:
Temos W =1H e VALOR=0
SUBWF VALOR,1 ; resultará W=0-1 = -1 C=0 (-)

Ex 2:
Temos W = 0H e VALOR=FFH
SUBWF VALOR,1 ; resultará W=ff-0 = ff C=1 (+)

SWAPF — **Swap Nibbles in f**

Syntax: [*label*] SWAPF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f<3:0>) → (dest<7:4>),
(f<7:4>) → (dest<3:0>)

Status Affected: None

Encoding:

00	1110	dfff	ffff

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0, the result is placed in W register. If 'd' is 1, the result is placed in register 'f'.

Words: 1

Cycles: 1

Example

```
SWAPF REG,  0
```

Before Instruction

REG1 = 0xA5

After Instruction

REG1 = 0xA5
W = 0x5A

Esta instrução troca os nibles de f. O registro de f tem seus bits trocados do 7 ao 4 trocados pelos do 3 ao 0.

Se d= 1 o resultado será salvo em f.
Se d= 0 o resultado será salvo em W.

Ex:
Temos VALOR = 25H

```
SWAPF   VALOR,0   ; resultará VALOR = 52H
```

XORLW — **Exclusive OR Literal with W**

Syntax: [*label*] XORLW k

Operands: 0 ≤ k ≤ 255

Operation: (W) .XOR. k → (W)

Status Affected: Z

Encoding:

11	1010	kkkk	kkkk

Description: The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example:

```
XORLW   0xAF
```

Before Instruction

W = 0xB5

After Instruction

W = 0x1A

Esta instrução XOR entre o conteúdo de W e "f".

Se d= 1 o resultado será salvo em f.
Se d= 0 o resultado será salvo em W.

Ex:
Temos VALOR = 52H
W= 52H

```
XORWF   VALOR,0   ; resultará em W = 0
```

XORWF	Exclusive OR W with f
Syntax:	[*label*] XORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .XOR. (f) → (dest)
Status Affected:	Z
Encoding:	00 | 0110 | dfff | ffff
Description:	Exclusive OR the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	XORWF REG 1 Before Instruction REG = 0xAF W = 0xB5 After Instruction REG = 0x1A W = 0xB5

Esta instrução XOR entre o conteúdo de W e o conteúdo de uma constante K.

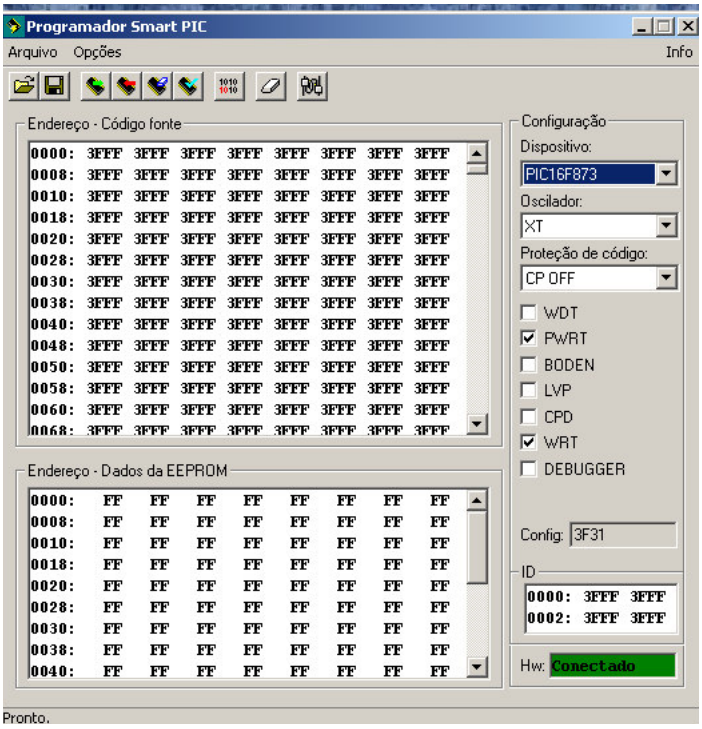
Ex:
Temos W= 10101101

XORLW 3AH ; resultará em W = 97H

Manual de Utilização do SMART PIC

Capítulo 6

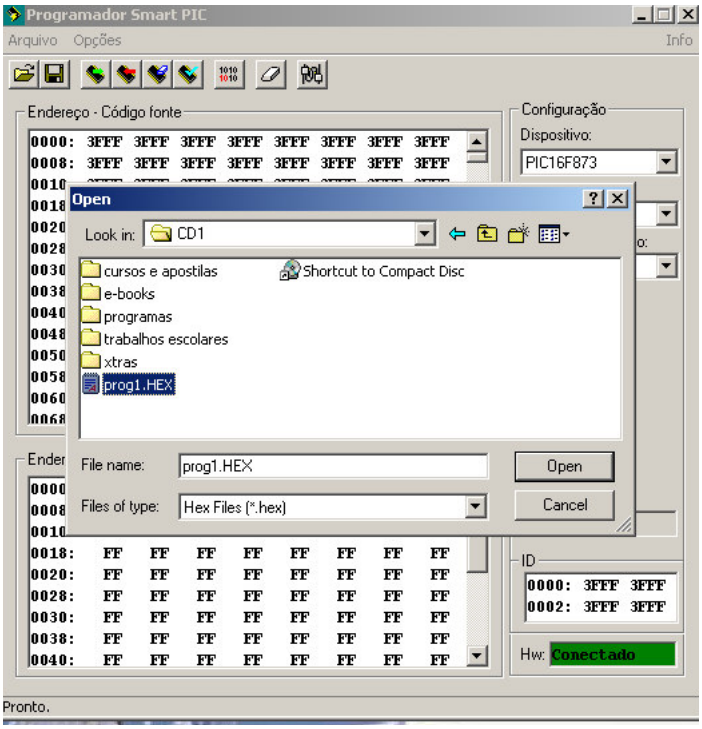
6 – Ao abrir o Programador Smart PIC selecione o PIC correto para a gravação em Dispositivo na tela principal.


Programador Smart PIC
Arquivo
Opções
Info
Endereço - Código fonte
0000: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0008: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0010: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0018: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0020: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0028: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0030: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0038: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0040: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0048: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0050: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0058: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0060: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0068: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
Endereço - Dados da EEPROM
0000: FF FF FF FF FF FF FF FF
0008: FF FF FF FF FF FF FF FF
0010: FF FF FF FF FF FF FF FF
0018: FF FF FF FF FF FF FF FF
0020: FF FF FF FF FF FF FF FF
0028: FF FF FF FF FF FF FF FF
0030: FF FF FF FF FF FF FF FF
0038: FF FF FF FF FF FF FF FF
0040: FF FF FF FF FF FF FF FF
Configuração
Dispositivo:
PIC16F873
Oscilador:
XT
Proteção de código:
CP OFF
WDT
PWRT
BODEN
LVP
CPD
WRT
DEBUGGER
Config: 3F31
ID
0000: 3FFF 3FFF
0002: 3FFF 3FFF
Hw: Conectado
Pronto.

Manual de Utilização do SMART PIC

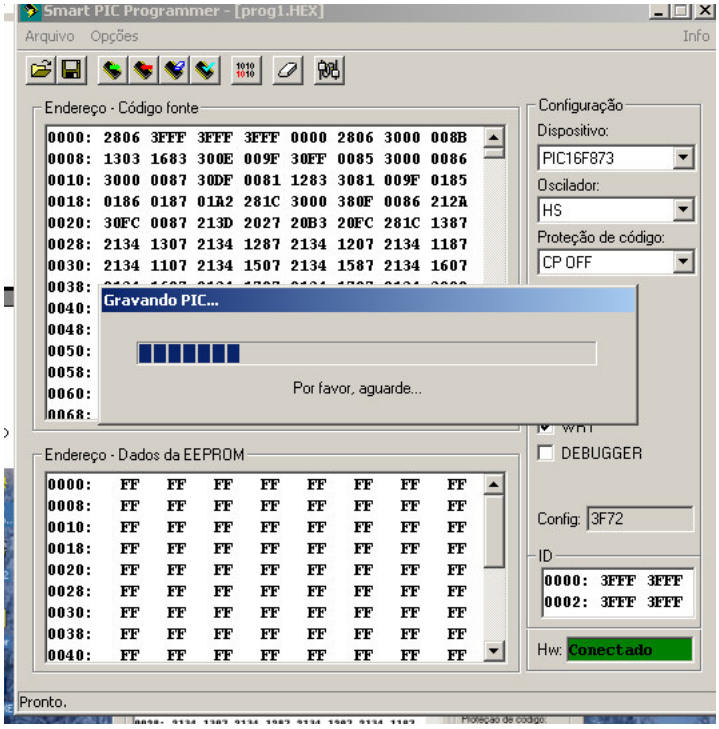
6.2 – Abrindo um . HEX para a gravação.

Clique em Arquivo-> Abrir e selecione o diretório que contenha o arquivo que deverá ser gravado no PIC. Após isto clique em OPEN.



Manual de Utilização do SMART PIC

6.2 – Após o programa ter sido carregado clique em Opções e em gravar o PIC ou use a tecla de atalho.


Smart PIC Programmer - [prog1.HEX]
Arquivo
Opções
Info
Endereço - Código fonte
0000: 2806 3FFF 3FFF 3FFF 0000 2806 3000 008B
0008: 1303 1683 300E 009F 30FF 0085 3000 0086
0010: 3000 0087 30DF 0081 1283 3081 009F 0185
0018: 0186 0187 01A2 281C 3000 380F 0086 212A
0020: 30FC 0087 213D 2027 20B3 20FC 281C 1387
0028: 2134 1307 2134 1287 2134 1207 2134 1187
0030: 2134 1107 2134 1507 2134 1587 2134 1607
Gravando PIC...
Por favor, aguarde...
Endereço - Dados da EEPROM
0000: FF FF FF FF FF FF FF FF
0008: FF FF FF FF FF FF FF FF
0010: FF FF FF FF FF FF FF FF
0018: FF FF FF FF FF FF FF FF
0020: FF FF FF FF FF FF FF FF
0028: FF FF FF FF FF FF FF FF
0030: FF FF FF FF FF FF FF FF
0038: FF FF FF FF FF FF FF FF
0040: FF FF FF FF FF FF FF FF
Configuração
Dispositivo:
PIC16F873
Oscilador:
HS
Proteção de código:
CP OFF
DEBUGGER
Config: 3F72
ID
0000: 3FFF 3FFF
0002: 3FFF 3FFF
Hw: Conectado
Pronto.

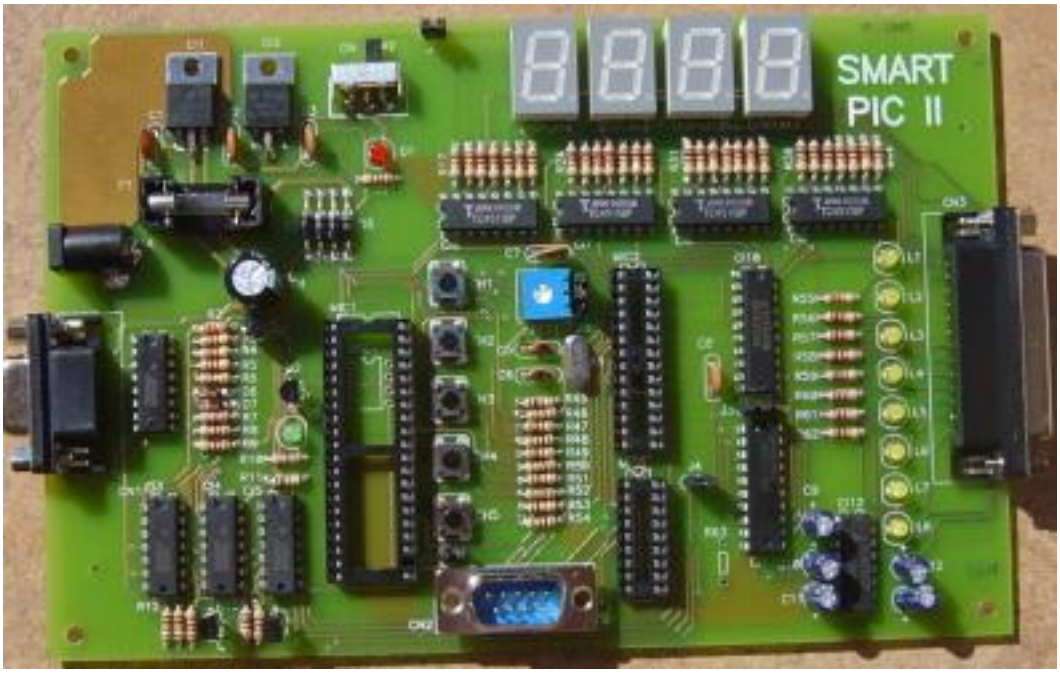
Após a gravação aparecerá a mensagem de “PIC programado com sucesso”

Manual de Utilização do SMART PIC

Capítulo 7

7 . Conhecendo a placa.

Ao ligar a chave principal o LED D1 deverá permanecer aceso.



- Quando estiver programando o LED D8 deverá estar aceso.
- Os PICs 16F84, 16F84A, 16F628, devem ser conectados ao ponto MIC3.
- Os PICs 16F873 e 16F873A, devem ser conectados ao ponto MIC2.
- Os PICs 16F877 e 16F877A, devem ser conectados ao ponto MIC1.
- NUNCA LIGUE DOIS UMA MAIS PICs SIMULTANEAMENTE.
- Os PICs 16F84, 16F877 e 16F877A apenas gravam, as entradas e saídas não funcionam com estes PICs.

Manual de Utilização do SMART PIC

7.1 - Configuração das PORTAS de entrada e saída para o PIC 16F84/A.

PORT	PINO	Configuração
RA0	17	Entrada Digital BT2 (CH2)
RA1	18	Entrada Digital BT1 (CH1)
RA2	01	Entrada Digital BT3 (CH3)
RA3	02	Entrada Digital BT4 (CH4)
RA4	03	Saída Digital LED1 (D9)
RB0	06	Entrada Digital BT5 (CH5) está no INT (interrupção) do 16F84/A
RB1	07	Saída Digital LED4 (D12)
RB2	08	Saída Digital LED3 (D11)
RB3	09	Saída Digital LED2 (D10)
RB4	10	Saída Digital LED5 (D13)
RB5	11	Saída Digital LED6 (D14)
RB6	12	Saída Digital LED7 (D15)
RB7	13	Saída Digital LED8 (D16)

7.2 - Configuração das PORTAS de entrada e saída para o PIC 16F873/A.

PORT	PINO	Configuração
RA0	02	Entrada Analógica
RA1	03	Entrada Digital BT1 (CH1)
RA2	04	Entrada Digital BT2 (CH2)
RA3	05	Entrada Digital BT3 (CH3)
RA4	06	Entrada Digital BT4 (CH4)
RA5	07	Entrada Digital BT5 (CH5)
RB0	21	Saída seleção display 1
RB1	22	Saída seleção display 2
RB2	23	Saída seleção display 3
RB3	24	Saída seleção display 4
RB4	25	Saída BCD para Display
RB5	26	Saída BCD para Display
RB6	27	Saída BCD para Display
RB7	28	Saída BCD para Display
RC0	11	Saída Digital LED8 (D16)
RC1	12	Saída Digital LED7 (D15)
RC2	13	Saída Digital LED6 (D14)
RC3	14	Saída Digital LED5 (D13)
RC4	15	Saída Digital LED4 (D12)
RC5	16	Saída Digital LED3 (D11)
RC6	17	Saída Digital LED2 (D10)
RC7	18	Saída Digital LED1 (D9)

Se você quer enviar para o display1 o valor 2 em decimal, selecione o display através dos bits RB0 até RB3 (0001), e envie o valor 2 em binário através dos bits RB4 até RB7 (0010).

Resumo
(RB0 ....RB3):
0001 – seleciona display 1
0010 – seleciona display 2
0100 – seleciona display 3
1000 – seleciona display 4

RB4 ... RB7
0000 – envia 0 decimal para o display seleccionado
0001 – envia 1 decimal para o display seleccionado
0010 – envia 2 decimal para o display seleccionado
0011 – envia 3 decimal para o display seleccionado
e assim sucessivamente.

Manual de Utilização do SMART PIC

7.3 - Conector CN2 (conector de entradas remotas)

Nota importante: Quando for utilizar as entradas pelo conector DB9 retirar o jumper J1. Para utilizar as entradas remotas basta colocar uma chave NA (normal aberta) fechando GND com a entrada desejada.

Pin Out DB9	Função
1	GND
2	CH5
3	CH4
4	CH3
5	CH2
6	+5Vcc
7	Entrada Analógica (RA0 pino 2 do PIC 16F873/A) **Importante: nunca coloque neste pino uma tensão maior que +5Vcc.**
8	CCP (capture compare) pino 13 do16F873/A. Quando utilizar este pino retirar o jumper (J4), e então este pino se tornará uma entrada. **Importante: não se esqueça de configurar corretamente o PORT para entrada.**
9	CH1

7.4 - Conector CN3 (conector de saídas remotas e comunicação serial)

Pin Out DB25	PIC 16F84/A	PIC 16F873/A
1		RB0 pino 22
2		RB4 pino 25
3		RB0 pino 21
4		RB3 pino 24
5		RB2 pino 23
6	RA4 pino 3	RC0 pino 11
7	RB3 pino 9	RC1 pino 12
8	RB2 pino 8	RC2 pino 13
9	RB1 pino 7	RC3 pino 14
10	RB4 pino 10	RC4 pino 15
11	RB5 pino 11	RC5 pino 16
12	RB6 pino 12	RC6 pino 17
13	RB7 pino 13	RC7 pino 18
14		RB5 pino 26
15		RB6 pino 27
16		RB7 pino 28
17,18 e19		+5Vcc
20, 21,22 e 23		GND
24		RX – Comunicação serial RS232
25		TX – Comunicação serial RS232

7.5 - Configuração dos Jumpers

Jumper	Função	Posição
J1	Desabilita/ Habilita entradas digitais modo local. CH1 à CH5	Se J1 inserido seleciona modo local se não seleciona modo remoto.
J2	Seleciona entrada analógica local/ remota	Se inserido fechando os pinos mais acima analógica modo local. Se inserido fechando os pinos mais abaixo analógica modo remoto pelo conector CN2 (pino 7)
J3	Seleciona entrada para comunicação RS232 ou saída digital.	Se inserido mais a direita seleciona entrada para comunicação serial (RC7 pino 18 PIC 16F873/A). Se inserido mais à esquerda seleciona saída digital.
J4	Seleciona entrada capture compare	Se inserido, saída digital, senão seleciona como entrada para Capture Compare conector CN2 pino 8 .
J5	Desligar/ ligar o display 7 segmentos.	Se inserido liga o display 7 segmentos, senão o mantém desligado. **Importante: Sempre que for utilizar as saídas que comandam o display, para comutar cargas remotas, retire este jumper para não sobrecarregar o circuito.**

Manual de Utilização do SMART PIC

7.6 – Conexão do KIT com o computador

O cabo que acompanha o kit SMART PIC II deve ser conectado ao CN1 e o lado DB25 deve ser conectado na porta paralela do computador (figura abaixo).